

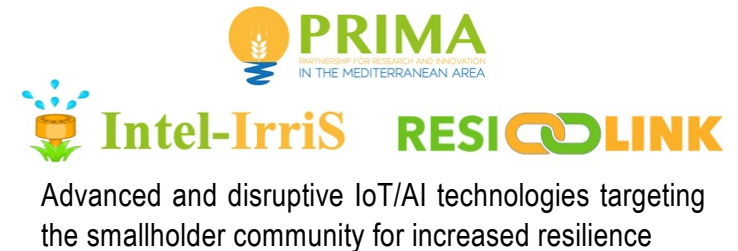
AI for Cyber-Physical Systems

Part 1 – Introduction

Prof. Congduc Pham
<http://www.univ-pau.fr/~cpham>



Horizon 2020
European Union funding
for Research & Innovation



Acknowledgements

- AI is probably one of the hot topics with the most incredible amount of also incredible quality contributions & tutorials videos!
- Setting up this course would not have been possible without all these contributors!
- I've crawled the web for hours (days), trying to take knowledge from multiple sources and trying to present all these impressive resources in a simple and comprehensive way for my students
- I tried to mention the sources of materials I've borrowed from the web as much as I could
- A special "thank you" for all the incredible tutorial videos from Machine Learnia (<https://machinelearnia.com/>, Guillaume Saint-Cirgue) and his YouTube channel (<https://www.youtube.com/@MachineLearnia/videos>)
- A special "thank you" akso for the complete and very clear videos from freeCodeCamp.org, especially on DL (<https://www.youtube.com/watch?v=dPWYUELwldM>)
- There are also all the articles from <https://towardsdatascience.com/>

The image features a cityscape at night, viewed from an elevated perspective. The sky is filled with several colorful, stylized clouds in shades of purple, red, orange, green, and blue. Each cloud contains white icons representing various IoT applications: a purple cloud with people and a heart rate monitor; a red cloud with a shopping cart, Euro symbol, and dollar sign; an orange cloud with a bridge, a bowl, and power towers; a green cloud with wind turbines and solar panels; a blue cloud with a car, a satellite, and a location pin; and a purple cloud with a factory, a truck, and gears. A light purple cloud contains a house and a server rack. Numerous arrows of various colors (purple, red, orange, blue, green) point upwards from the city towards the clouds, symbolizing the growth and deployment of IoT devices. The overall scene is vibrant and futuristic.

**2022, billions of IoT devices
are deployed worldwide!**

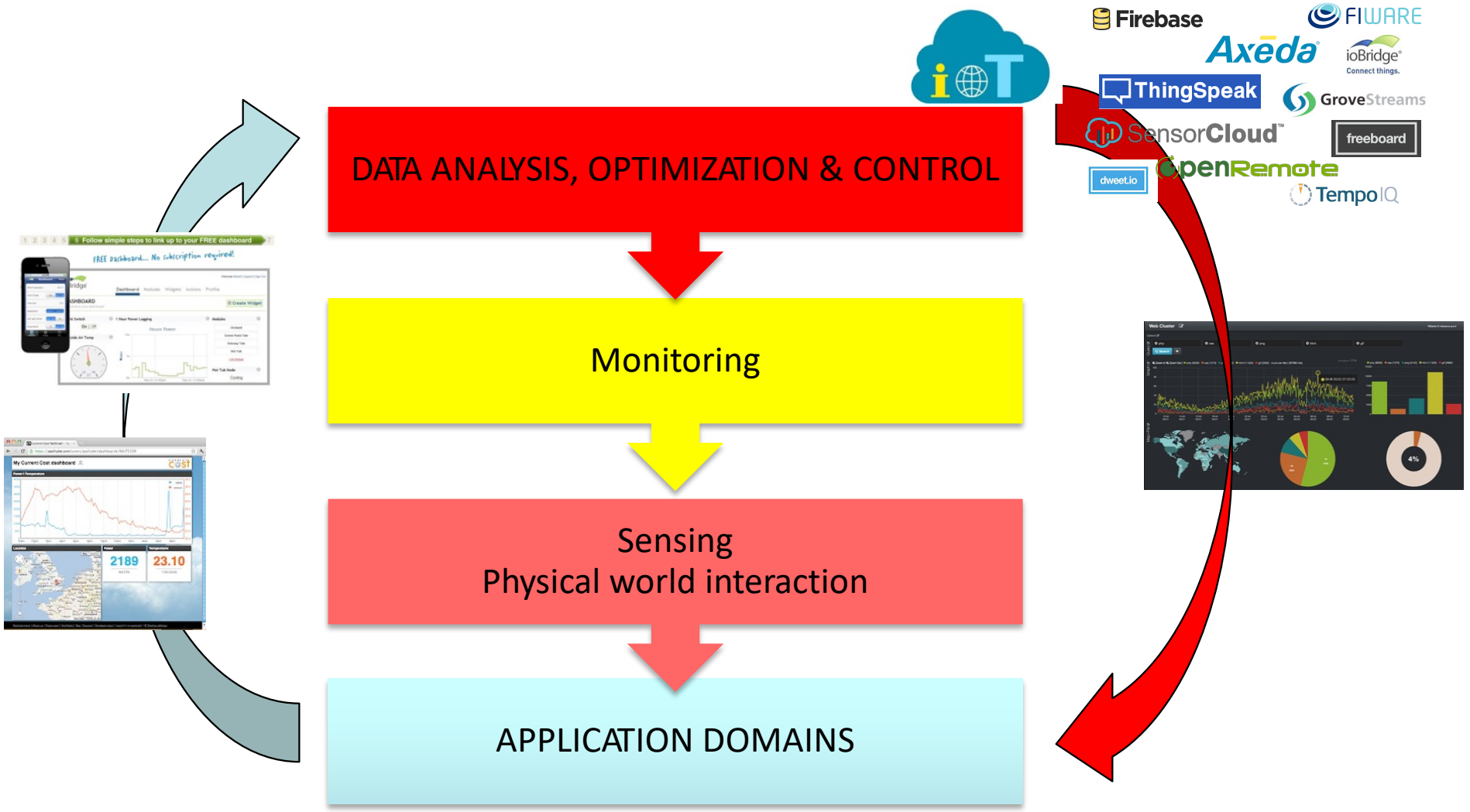
IoT added-values come from interactions & linked data!

Pr. Congduc Pham
<http://www.univ-pau.fr/~cpham>



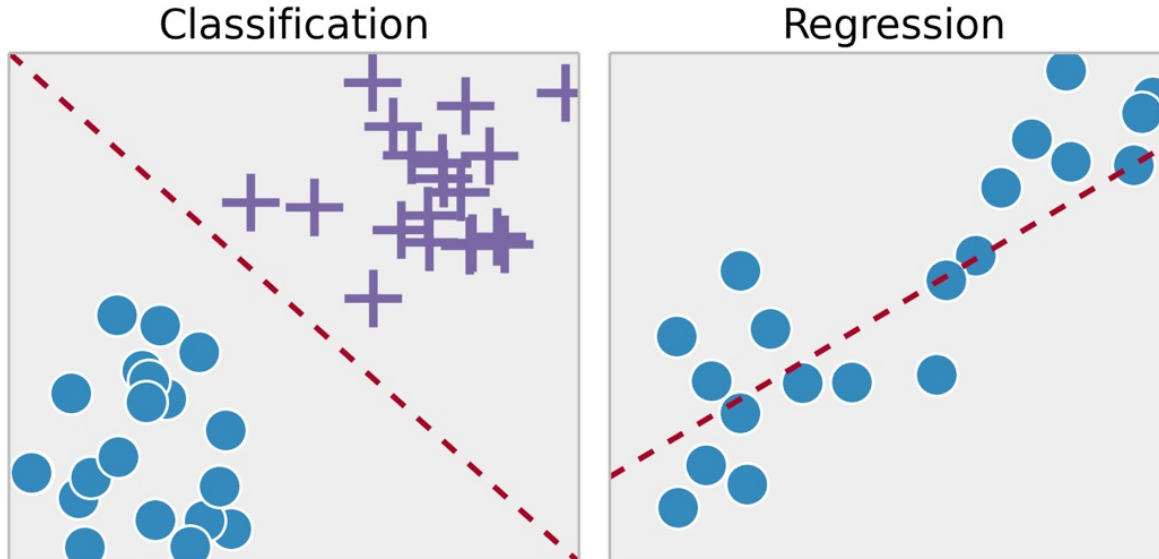
IoT to monitor, optimize & control...

Pr. Congduc Pham
<http://www.univ-pau.fr/~cpham>



...but also how to analyse the data

- ⦿ What is the meaning of the collected data?
 - ⦿ Classification, ...
- ⦿ Can we predict how the data will evolve?
 - ⦿ Prediction using Regression methods



Cyber-Physical Systems

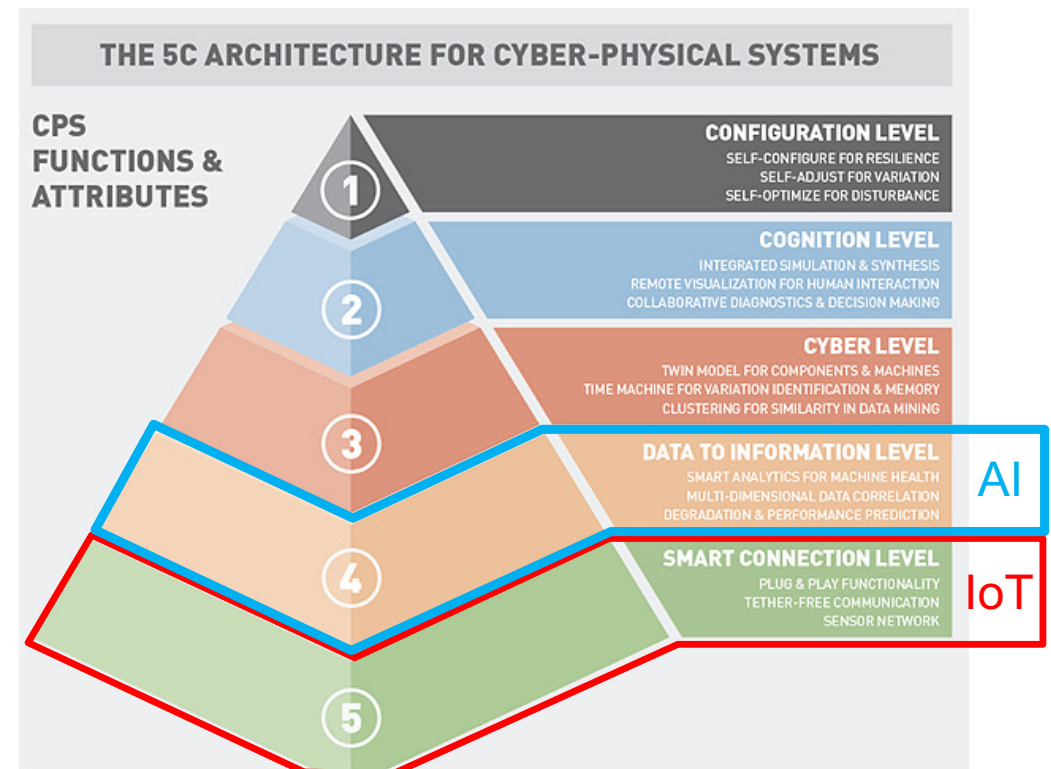
- ⦿ Norbert Wiener, early definition:
"science of control and communication in machines and humans"
- ⦿ From wikipedia:
"A cyber-physical system (CPS) or intelligent system is a computer system in which a mechanism is controlled or monitored by computer-based algorithms. In cyber-physical systems, physical and software components are deeply intertwined, able to operate on different spatial and temporal scales, exhibit multiple and distinct behavioral modalities, and interact with each other in ways that change with context"
- ⦿ From NSF:
"Cyber-physical systems integrate sensing, computation, control and networking into physical objects and infrastructure, connecting them to the Internet and to each other"

CPS & IoT

- ④ *"The concepts of Internet of Things (IoT) and Cyber Physical Systems (CPS) are closely related to each other. IoT is often used to refer to small interconnected devices like those in smart home while CPS often refers to large interconnected devices like industry machines and smart cars"*
- ④ *"Cyber Physical Systems represent a complex combination between physical and computational elements, and can be found in a multitude of areas such as manufacturing, aerospace, automotive, transportation, etc"*
- ④ *"Sensors, computational and physical world are closely integrated in CPS. CPS includes both conventional embedded systems and control systems that are presumed to be remodelled by emerging methodologies and integration of the Internet of Things. IoT is the base or enabling technology for cyber-physical systems."*
- ④ *"The intelligent Internet of Things (IoT) and cyber–physical systems (CPS) are evolving quickly as interdisciplinary technologies that combine physical components and computational devices to enable artificial intelligence (AI)-based solutions"*

Some advanced CPS-related concepts

- ⦿ Intelligent **robot systems**
- ⦿ Augmented Reality, **Virtual & Mixed Reality**
- ⦿ Real-time **multi-agent systems**
- ⦿ Cooperative **distributed simulations**
- ⦿ Coupling of **Real-time and Co-simulation**
- ⦿ **Digital Twins** to create high-fidelity virtual models of physical objects in order to simulate their behaviors in the real world and provide feedback





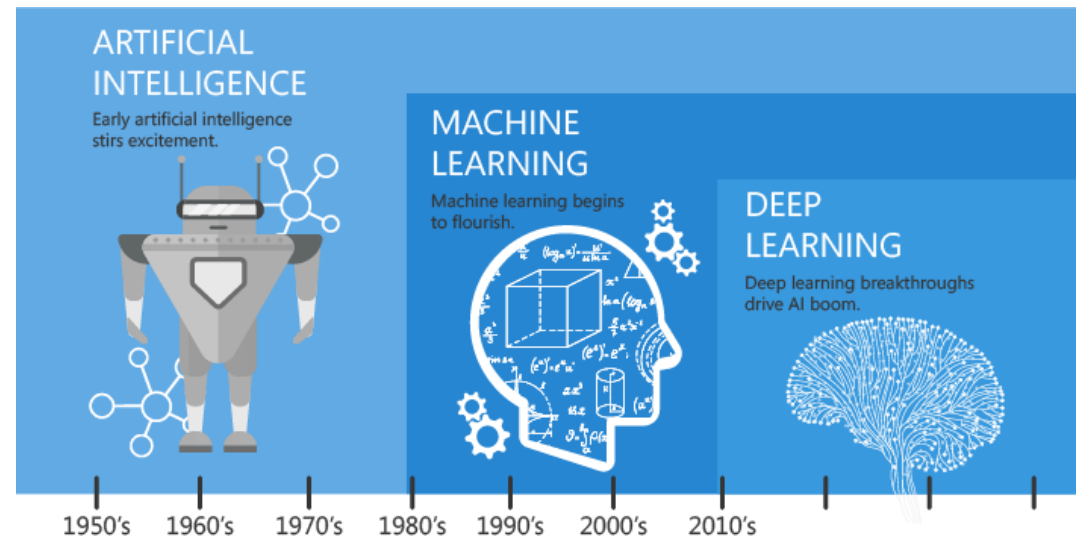
INTRO

TO

AI

The raise of Artificial Intelligence

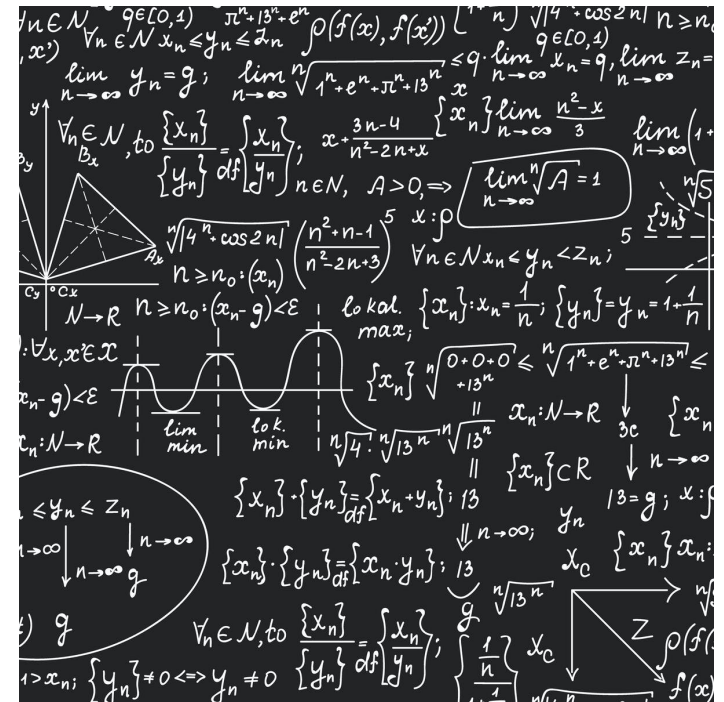
- ⦿ It is the science and engineering of making intelligent machines.
- ⦿ In Computer Science, Artificial Intelligence (AI) research is defined as the study of « intelligent agents »
- ⦿ From General AI to Narrow AI: from overhyping to fewer promises, but more realistic!



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

AI: (now) a serious science!

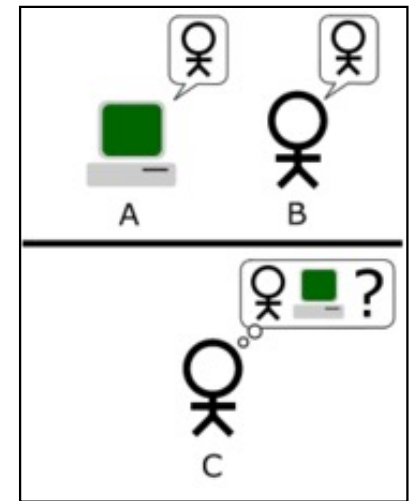
- ⦿ General-purpose AI like the robots of science fiction is incredibly hard
 - ⦿ Human brain appears to have lots of special and general functions, integrated in some amazing way that we really do not understand (yet)
- ⦿ Special-purpose AI is more doable (nontrivial)
 - ⦿ E.g., chess/poker playing programs, logistics planning, automated translation, speech and image recognition, web search, data mining, medical diagnosis, keeping a car on the road.



The Turing Test



- Proposed By Alan Turing in 1950
- To be called intelligent, a machine must produce responses that are indistinguishable from those of a human.
- Human judge communicates with a human and a machine over text-only channel.
- Both human and machine try to act like a human. Judge tries to tell which is which.
- Is Turing Test the right goal?



“Aeronautical engineering texts do not define the goal of their field as making ‘machines that fly so exactly like pigeons that they can fool even other pigeons.’”
[Russell and Norvig]

Reflection

if AI can be **more**
rational than
 humans in some
 cases, why not?

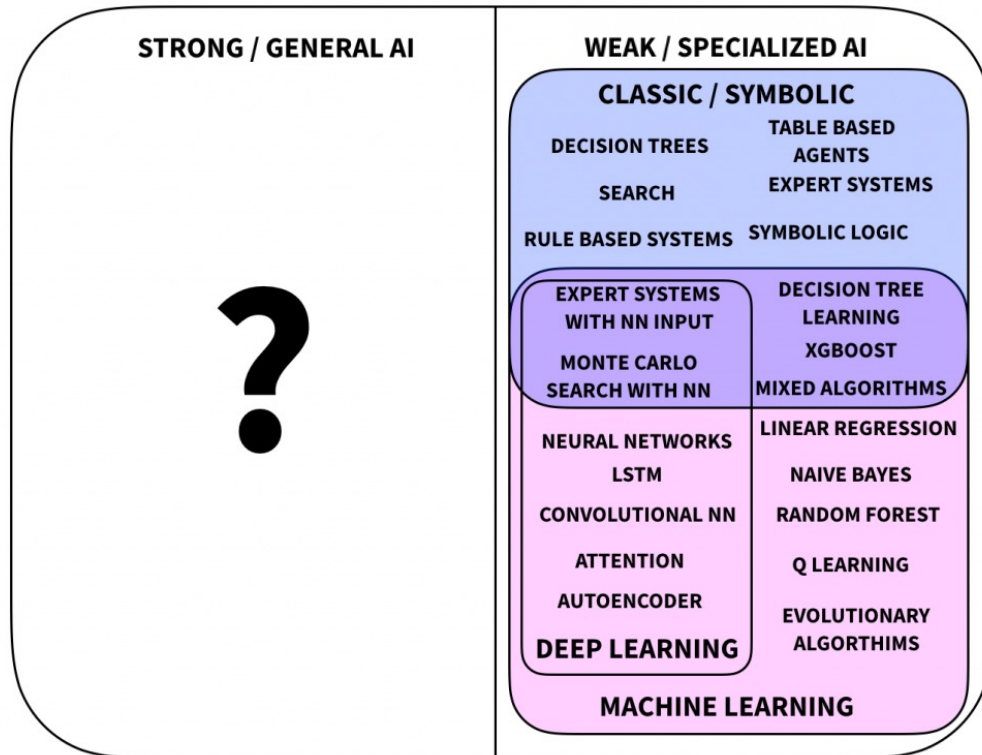


Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that <u>act</u> <u>rationally</u>

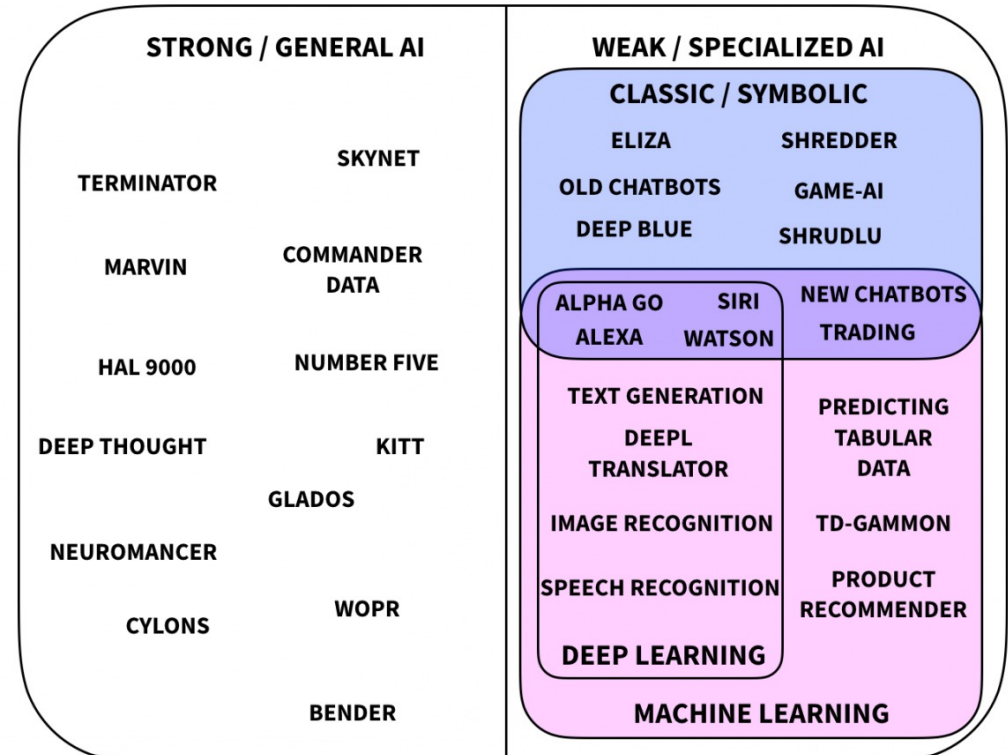
AI focus on **action**.
 Avoids philosophical
 issues such as “is the
 system conscious” etc.

AI Technologies

AI TECHNOLOGIES

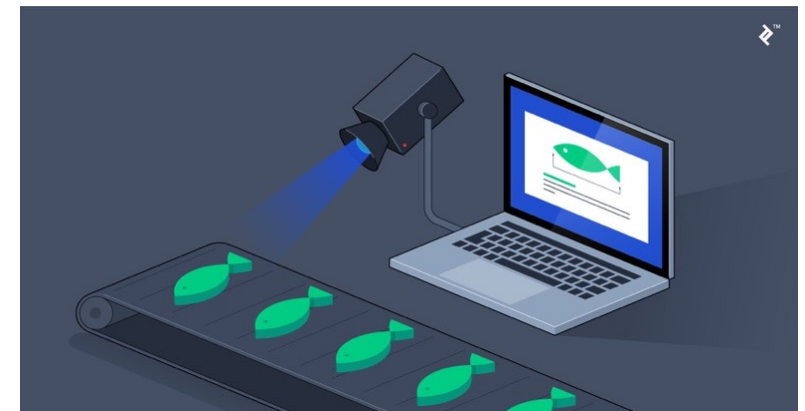
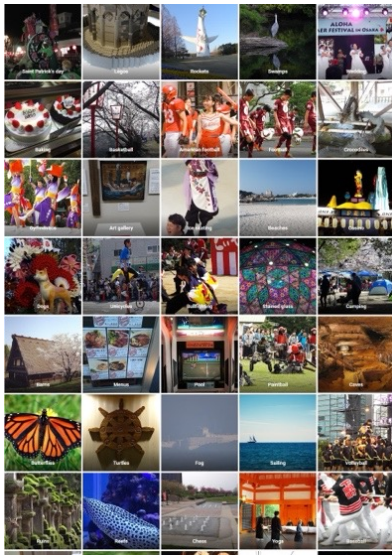
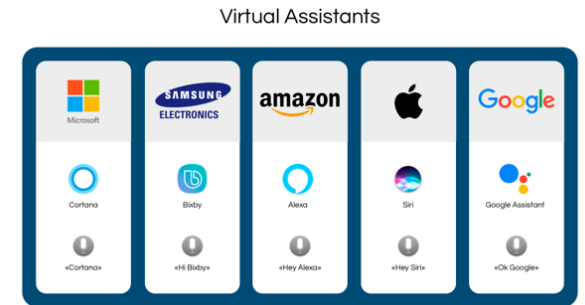
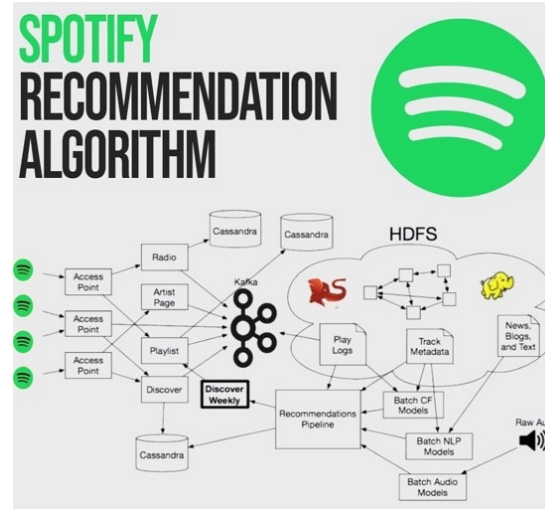
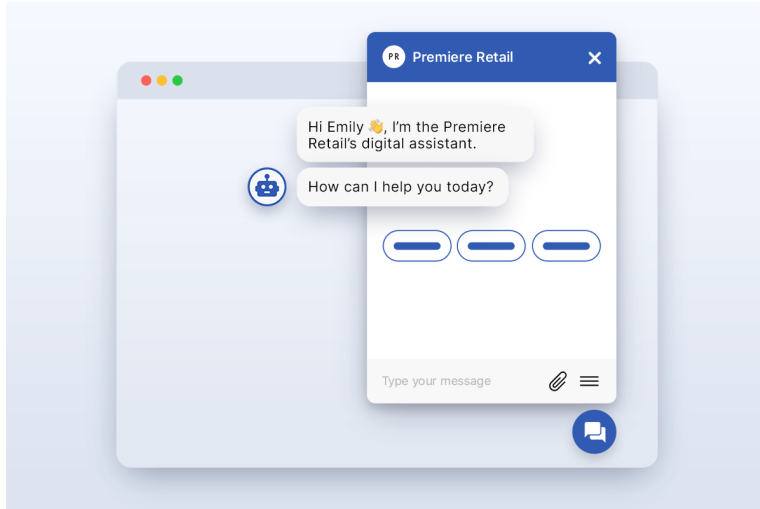


AI EXAMPLES



<https://divis.io/en/2019/03/ai-for-laymen-part-2-symbolic-ai-neural-networks-and-deep-learning/>

AI in our daily life



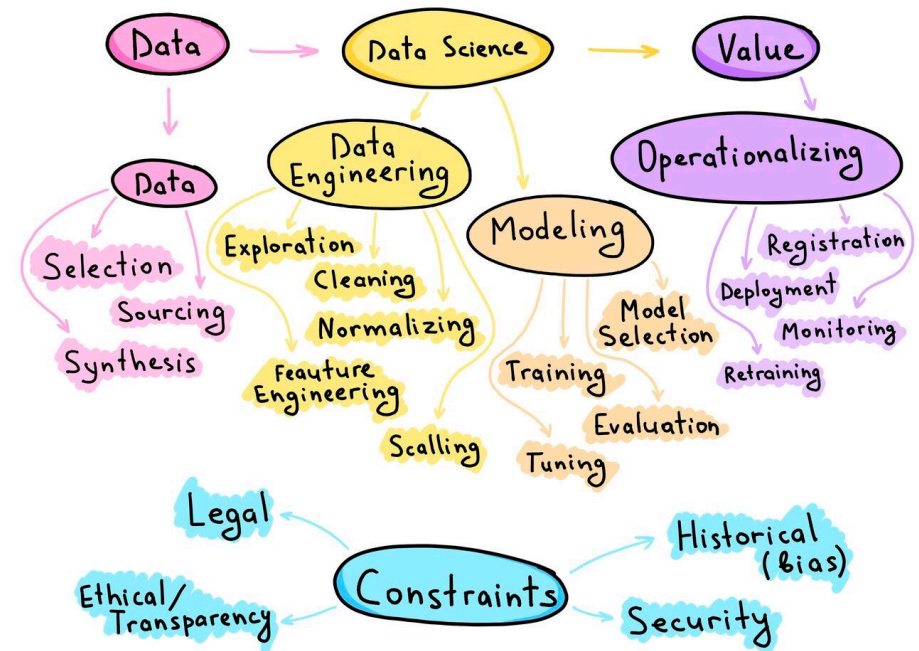
AI in industries

- ⦿ Infrastructure monitoring, Security & Safety
- ⦿ Continuous process improvement, Process automation, Process optimization
- ⦿ Smart logistics management, remote management, tracking,
- ⦿ Connectivity to back-end system, integration of smart tools, Interoperability
- ⦿ Data analysis, Supply Chain Optimization, Predictive maintenance

WHAT COMPANIES THINK A.I. LOOKS LIKE



WHAT IT ACTUALLY IS



AI in industries

- Infr
- Sec
- Cor
- imp
- aut
- opt
- Sm
- rem
- Cor
- sys
- too
- Dat
- Opt
- maintenance

WHAT COMPANIES

LIKE

IS

Value

Personalizing

Registration

Deployment

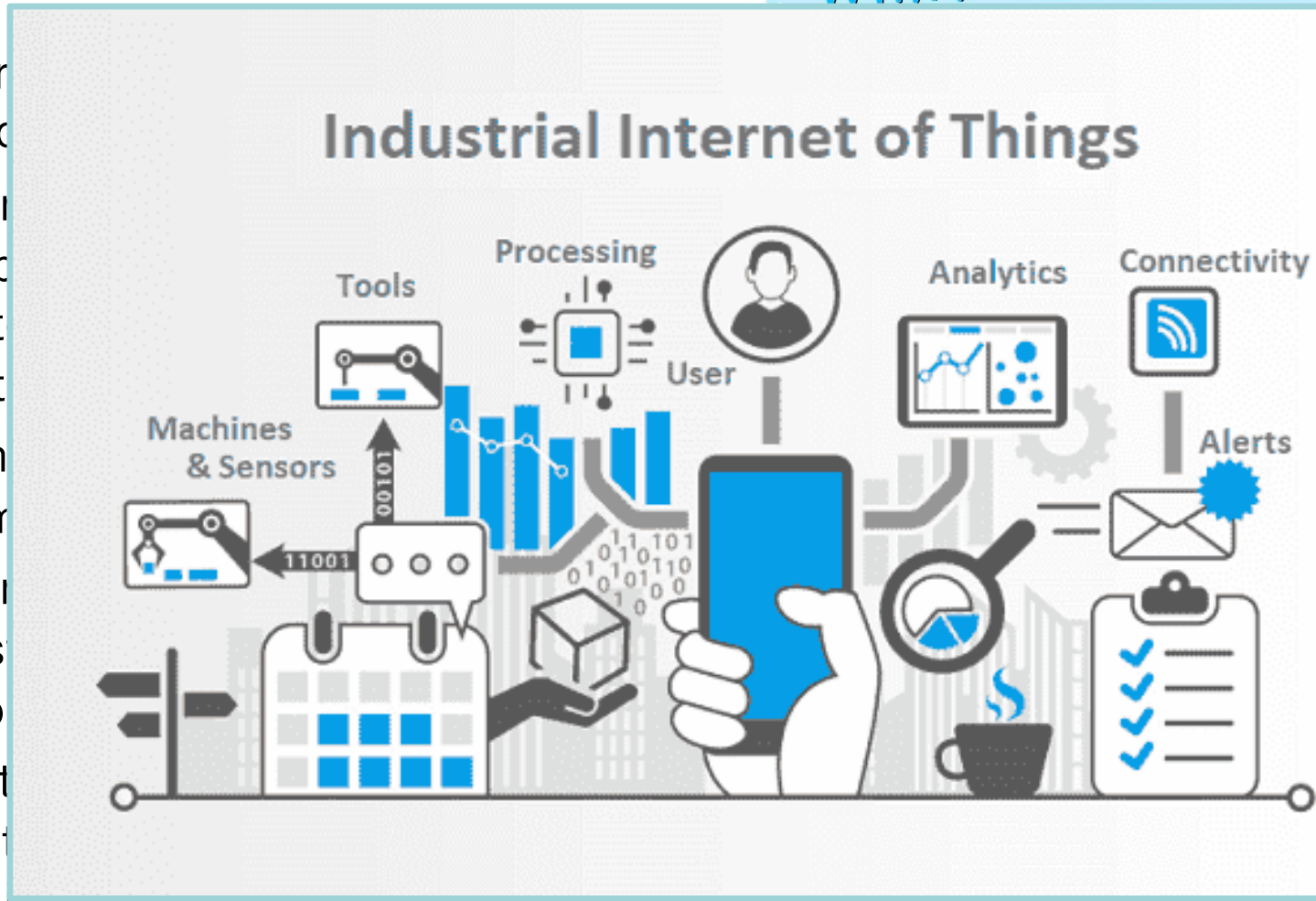
Monitoring

Retraining

Algorithmic (bias)

Security

Pr. Congduc Pham
<http://www.univ-pau.fr/~cpham>

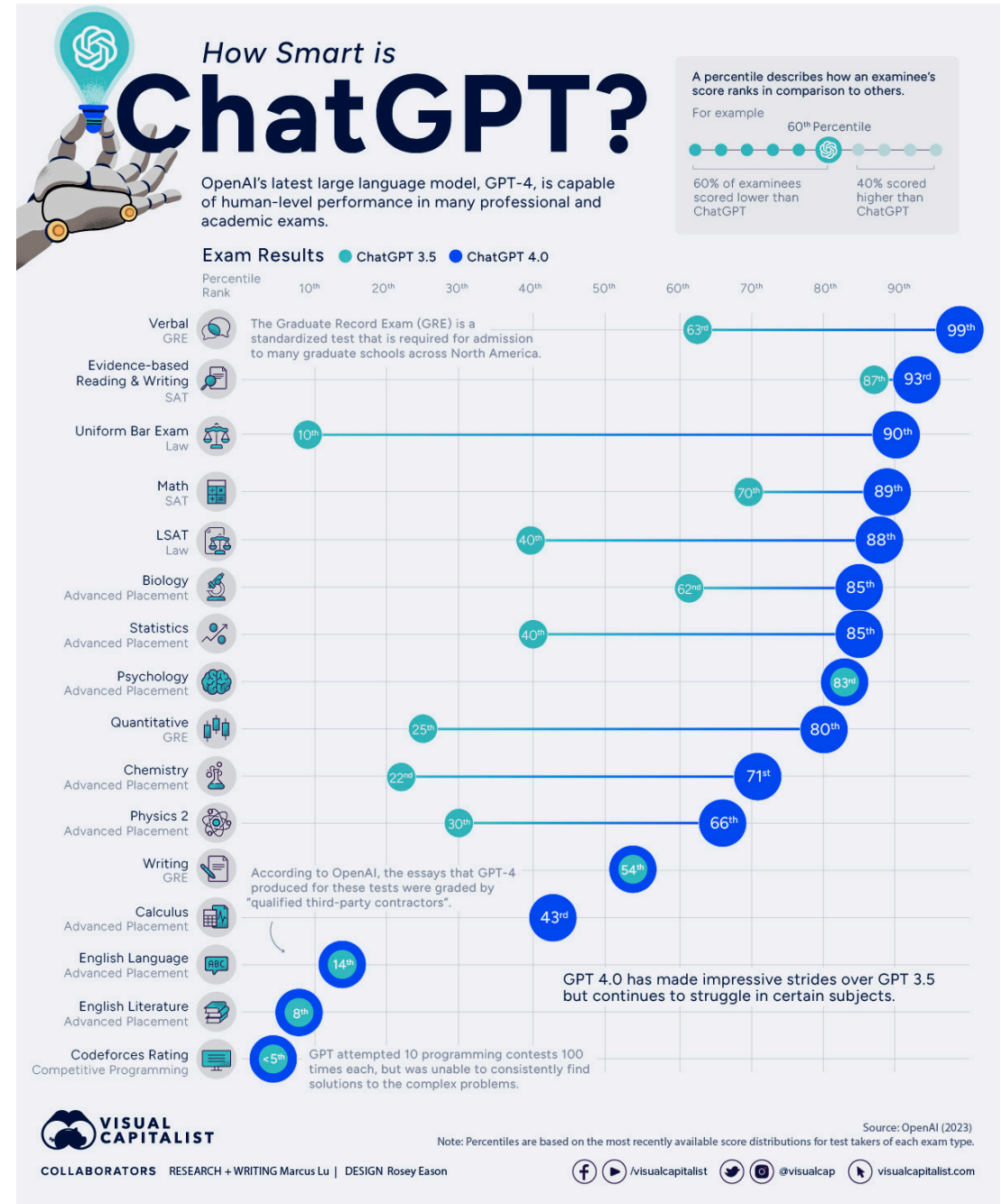


Transparency

Security

And ChatGPT?

- Chat Generative Pre-trained Transformer, is a chatbot that simulates human conversation through text responses
- ChatGPT can generate text, making it useful for content creation tasks such as writing articles, stories, and even poetry
- It is a great tool to quickly get reviews or articles on specific domains!

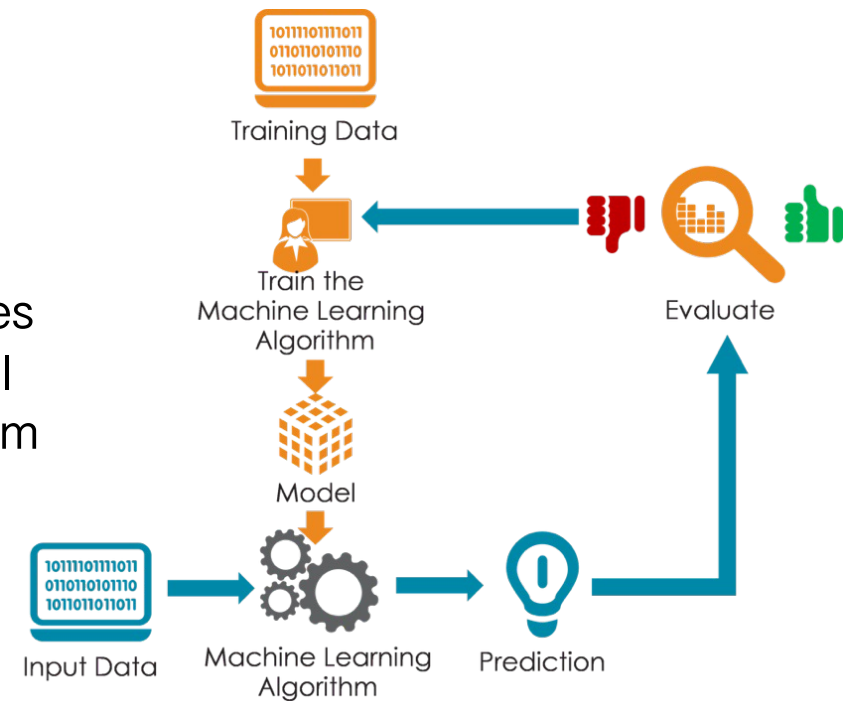


Let's try to be clear

- ⦿ The majority of **AI remains weak or narrow AI** and is only able to complete a single task
- ⦿ Most AI projects use Machine Learning techniques
- ⦿ Machine Learning focuses on teaching computers how to learn without being programmed to do specific tasks – **Deep Learning is a subset of Machine Learning!**
- ⦿ Deep Learning is differentiated from other types of machine learning based on how the algorithm learns and how much data the algorithm uses. **Deep Learning requires large data sets, but it needs minimal manual human intervention**
- ⦿ Deep Learning is intended to **mimic the structure of a human brain**, with complex, multi-layered neural networks

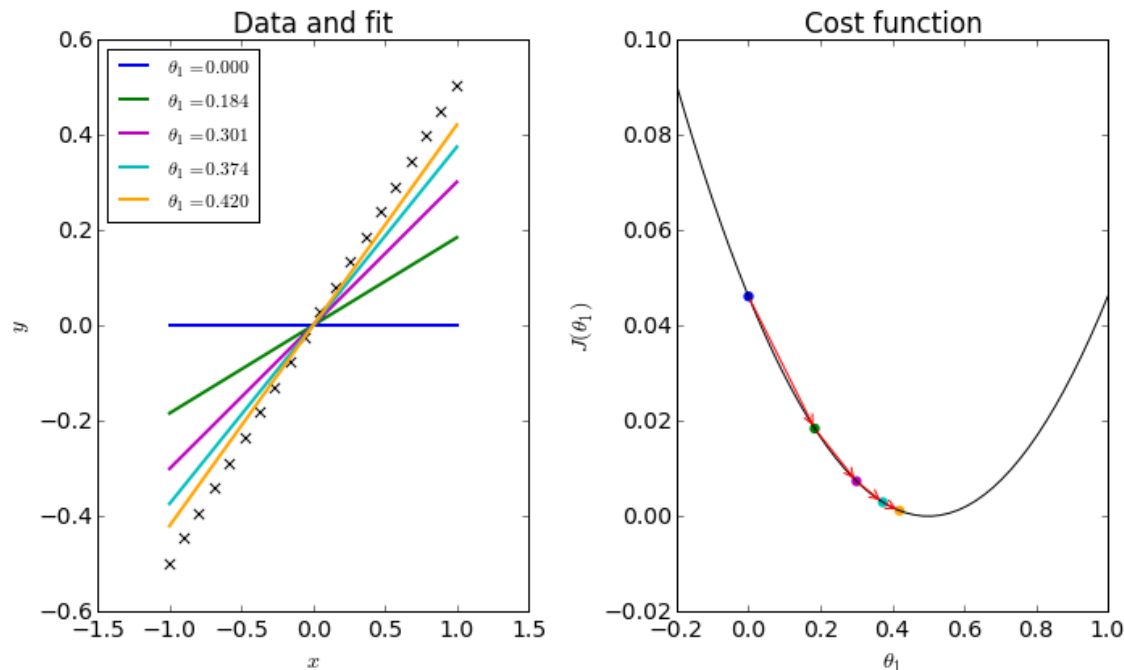
Machine Learning

- Develops Narrow Artificial Intelligence systems through examples
 - A developer creates a model and then “trains” it by providing it with many examples
 - The machine learning algorithm processes the examples and creates a mathematical representation of the data that can perform prediction and classification tasks
- Example
 - A machine-learning algorithm trained on thousands of emails with their category (legitimate or spam) will be able to predict if a new mail is spam or not



Optimization in AI

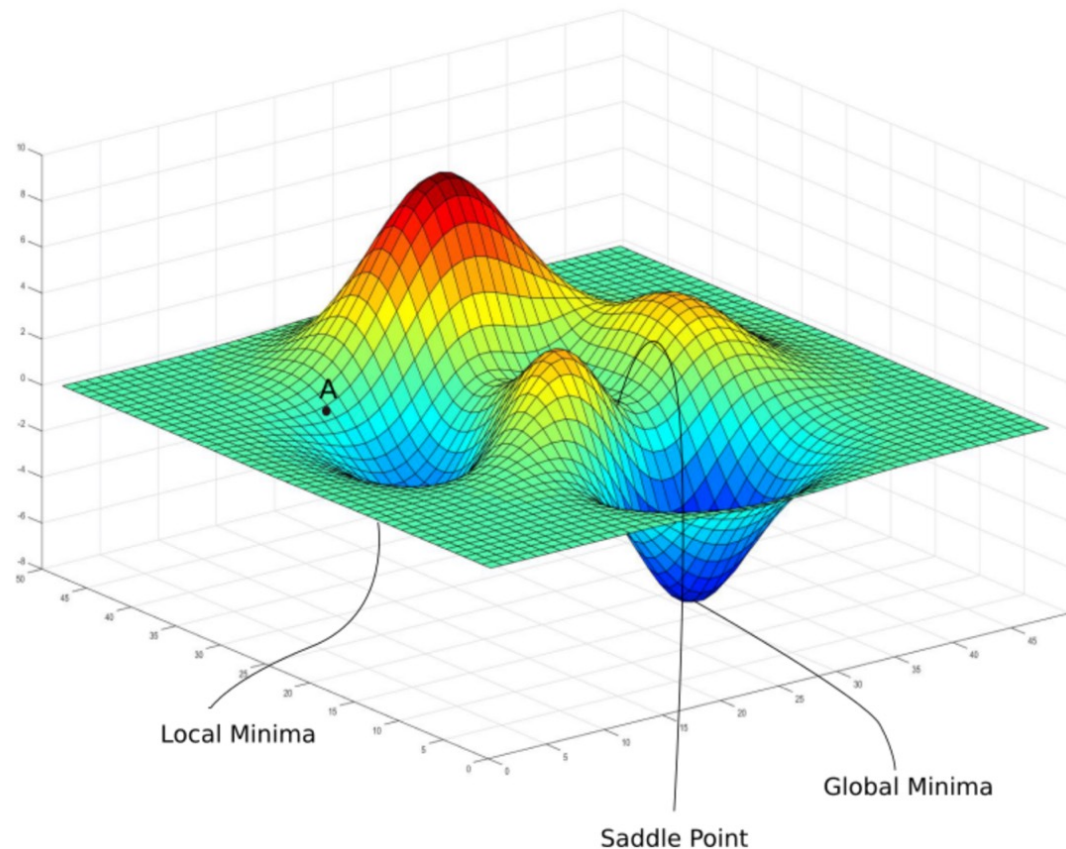
- Most AI problems consist of approximating a function that maps examples of inputs to examples of outputs
- Modern AI can be summarized as **optimizing a performance criterion using example data or past experience**



Pictures from <https://scipython.com/blog/visualizing-the-gradient-descent-method/>

Optimization can become complex!

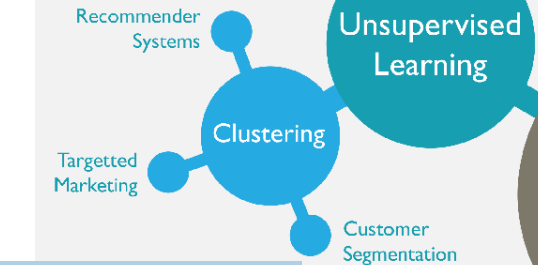
- ⦿ What if the inputs are very complex? Trying to find the best fitted function can become incredibly hard!



Machine Learning Techniques

Optimize a performance criterion using example data or past experience

A lot of statistic methods



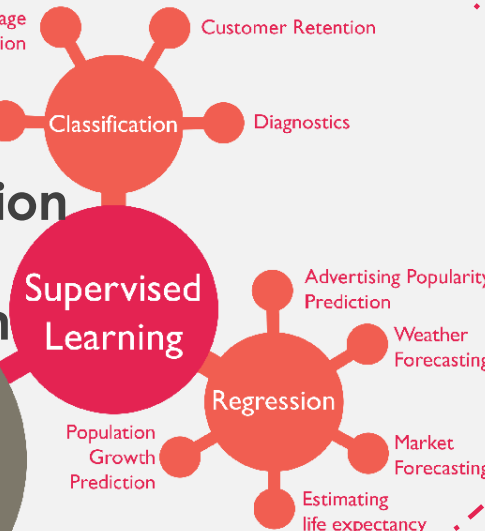
Role of Statistics: Inference from a sample

Role of Computer science: Efficient algorithms to (i) solve the optimization problem (ii) represent and evaluate the model for inference

Machine Learning Bubble Chart

Representation
Evaluation
Optimization

Machine Learning



Classification

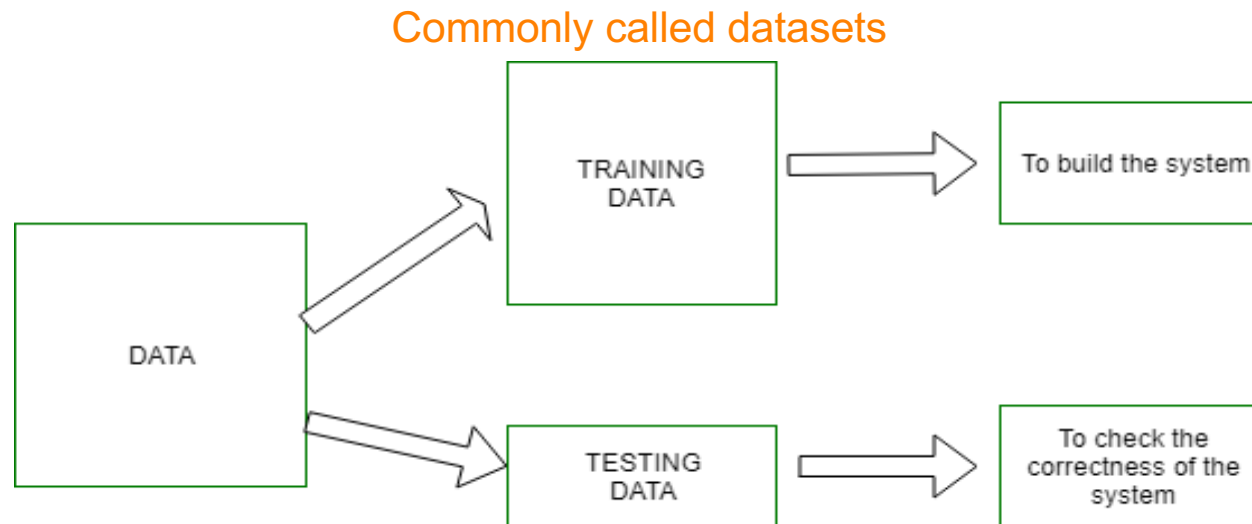
- Logistic Regression
- Support Vector Machine
- Naives Bayes
- K-Nearest Neighbor
- Decision Tree
- Random Forest

Regression

- Linear Regression
- Lasso Regression
- Ridge Regression
- Loess Regression
- Spline Regression

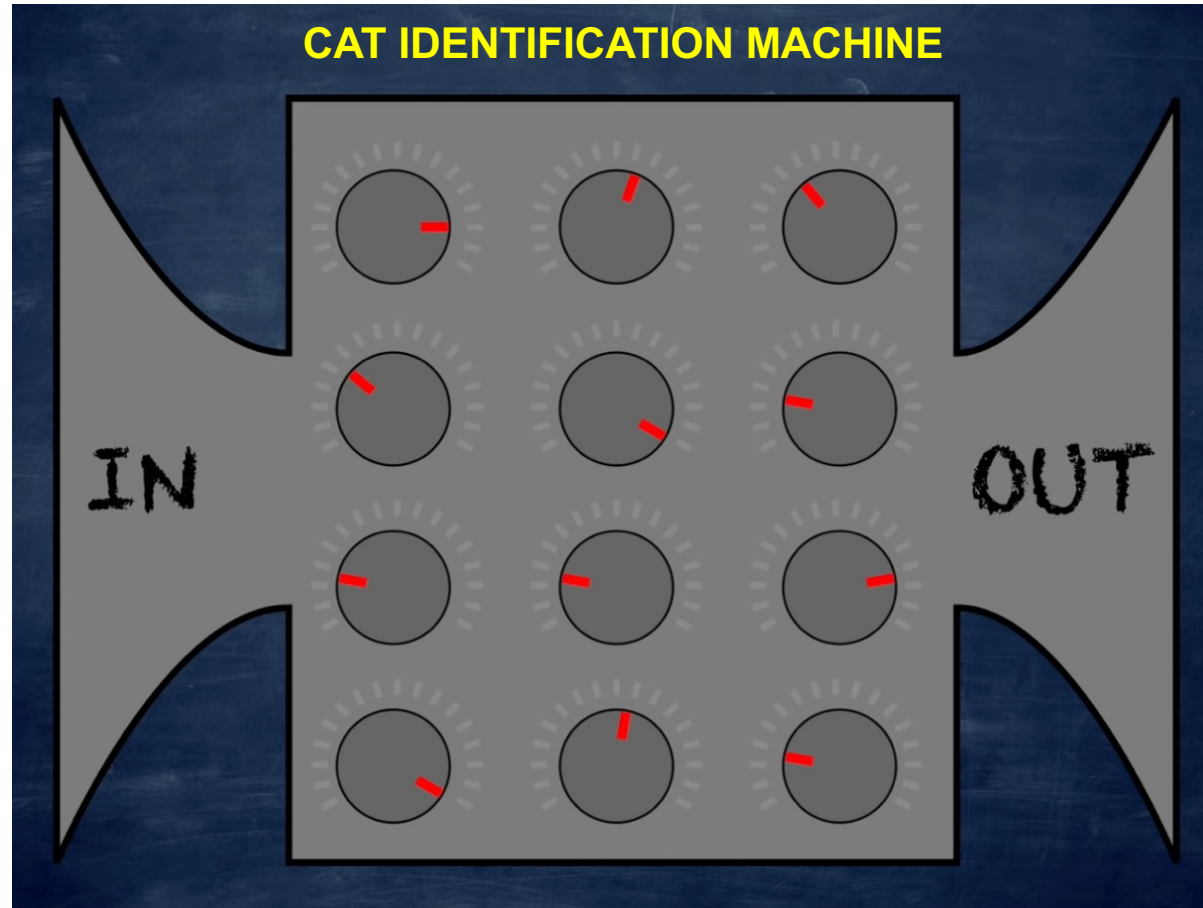
Supervised Learning

- ⦿ ML model is presented with *input data* which is **labeled**
 - ⦿ Each *input data* is tagged with the correct label
- ⦿ The goal is to approximate math operations in the ML model so well that when presented with new *input data*, the ML model can **predict** the output variables for that new *input data*



Supervised learning- an alternative view

CAT!

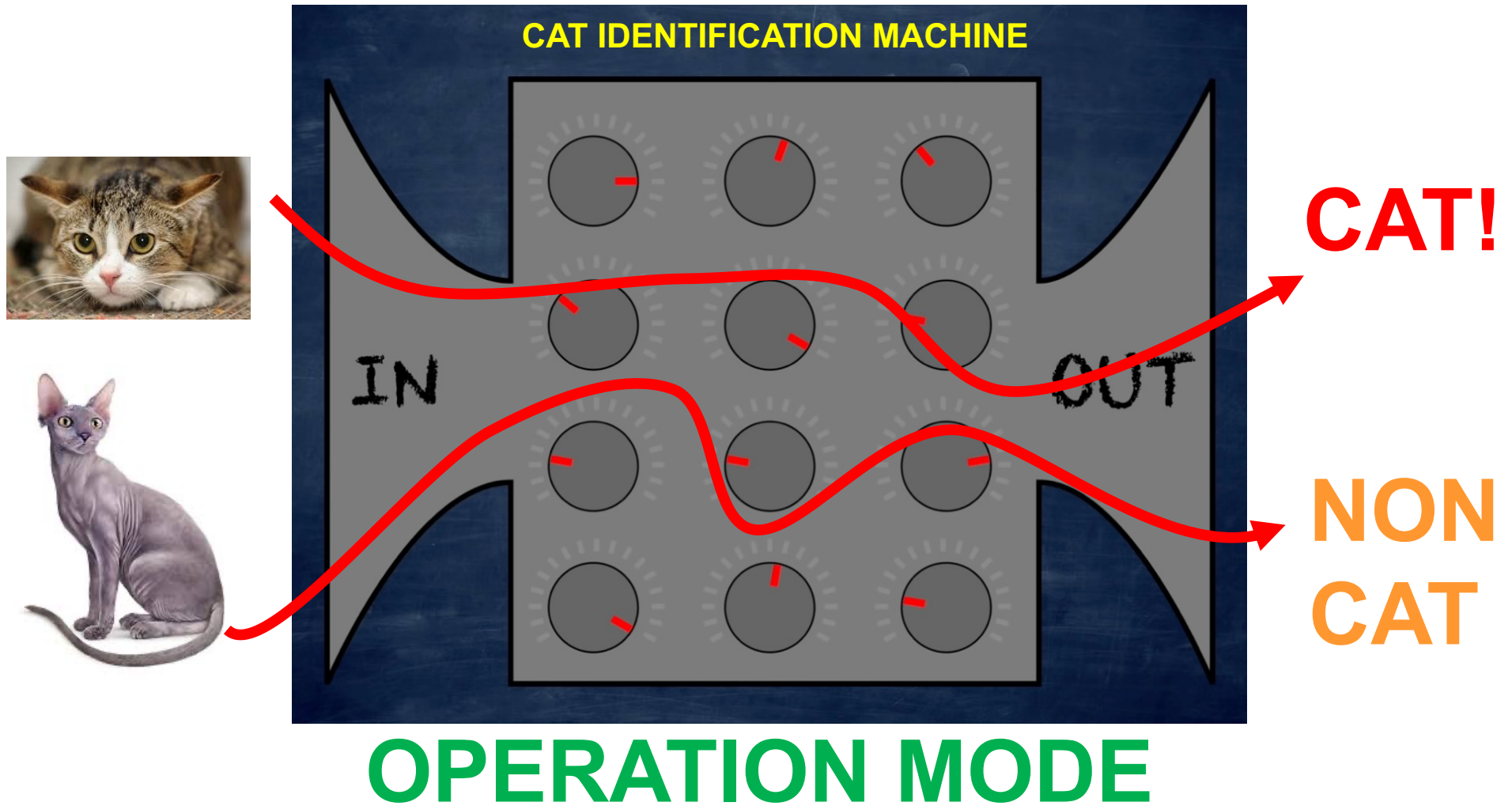


CAT

**NON
CAT**

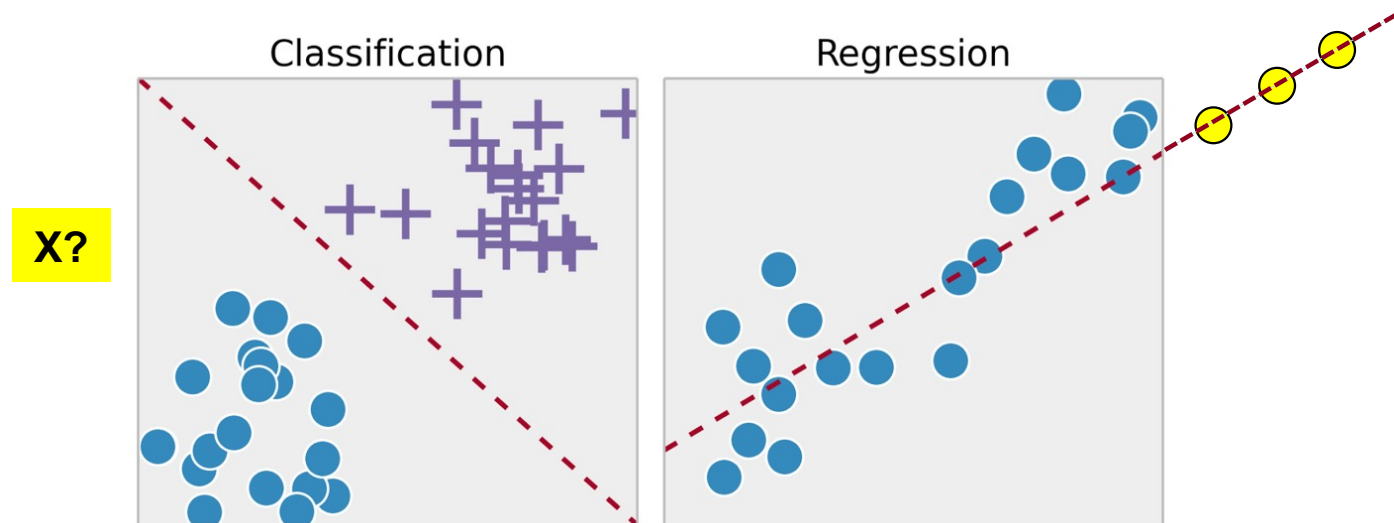
TRAINING MODE

Supervised learning- an alternative view



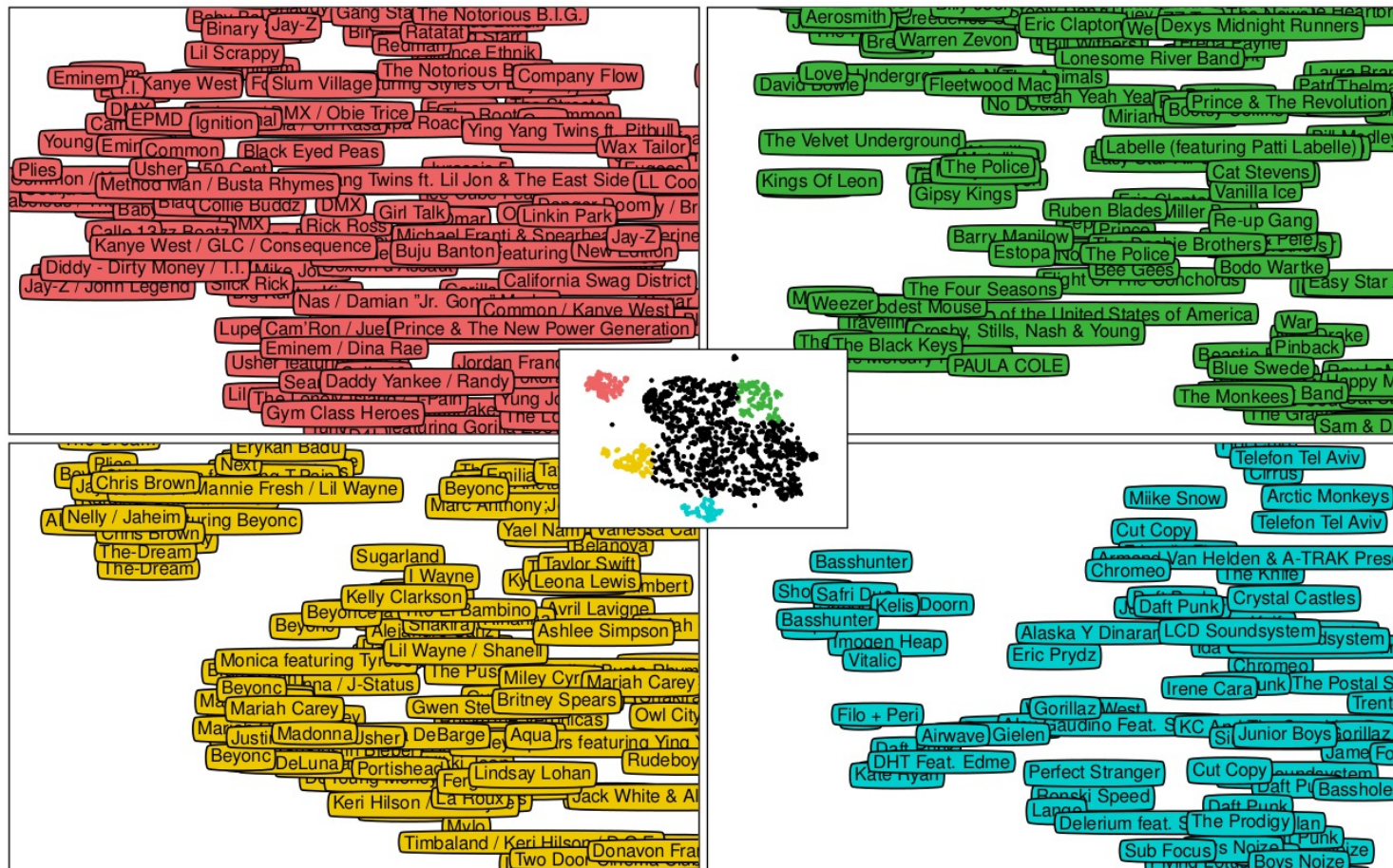
Classification vs Regression

- ⦿ **Classification:** A classification problem is when the output is a category, such as “red” or “blue” or “disease” and “no disease”.
- ⦿ **Regression:** A regression problem is when the output is a real number, such as “dollars” or “weight”.



Classification in real-world problem

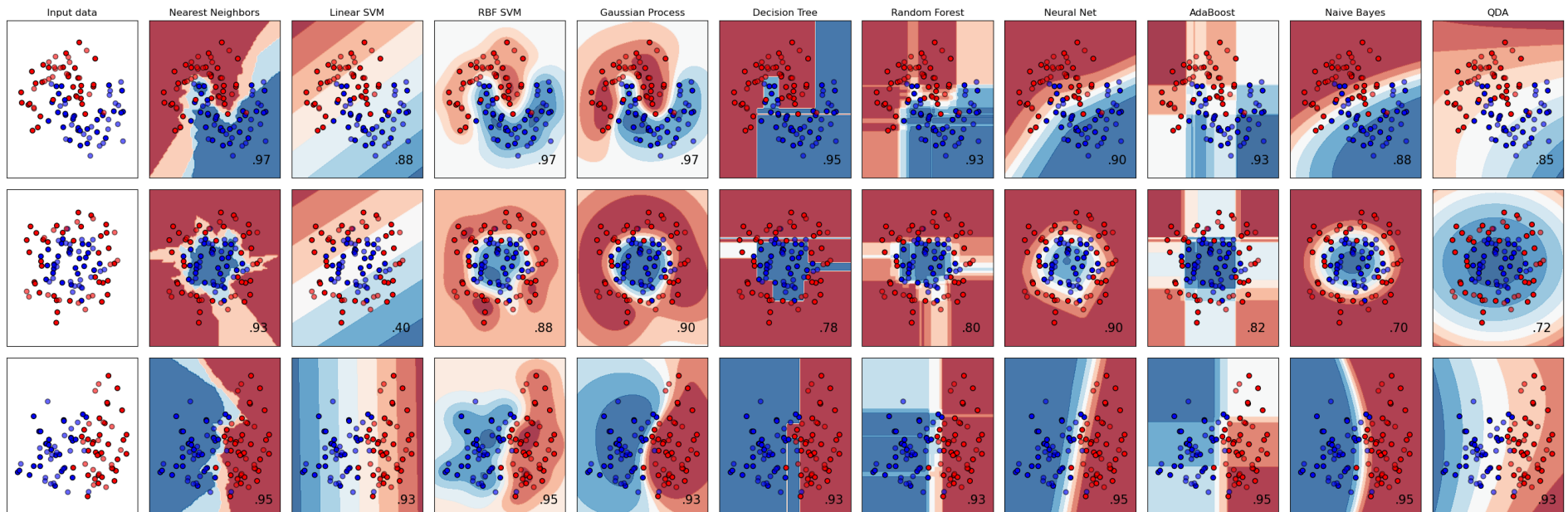
◉ <https://sander.ai/2014/08/05/spotify-cnns.html>



Classification methods?

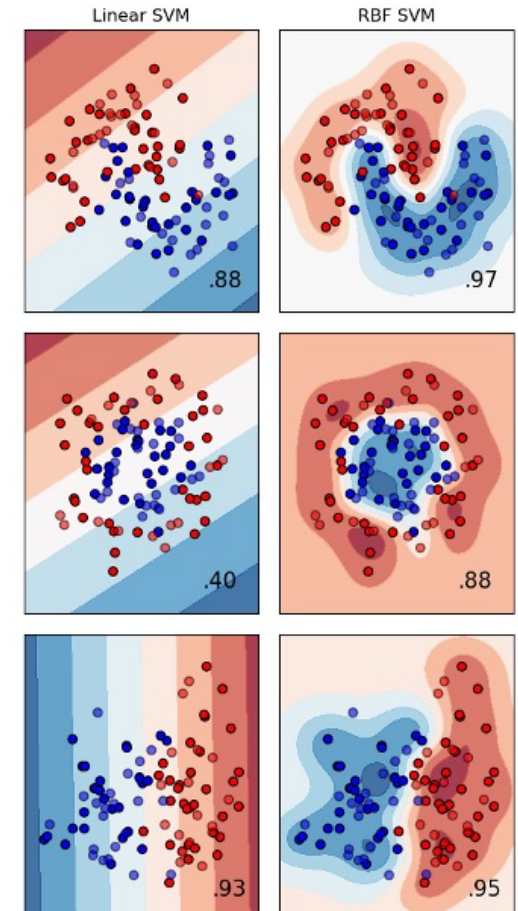
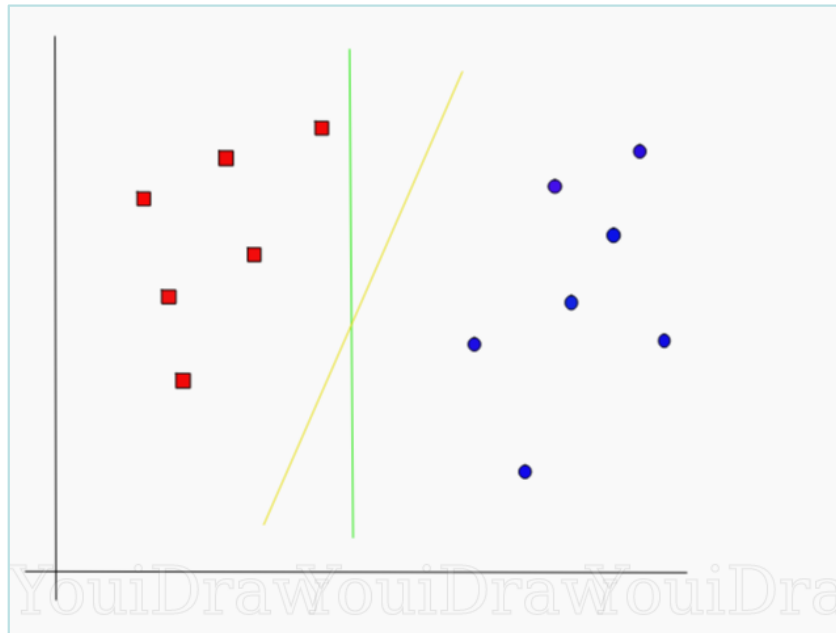
- ⦿ This is where the choice of a classification method is important: linear/nonlinear, categorical/continuous, supervised/non-supervised,...

- Classification
 - Logistic Regression
 - Support Vector Machine
 - Naives Bayes
 - K-Nearest Neighbor
 - Decision Tree
 - Random Forest
 - ...



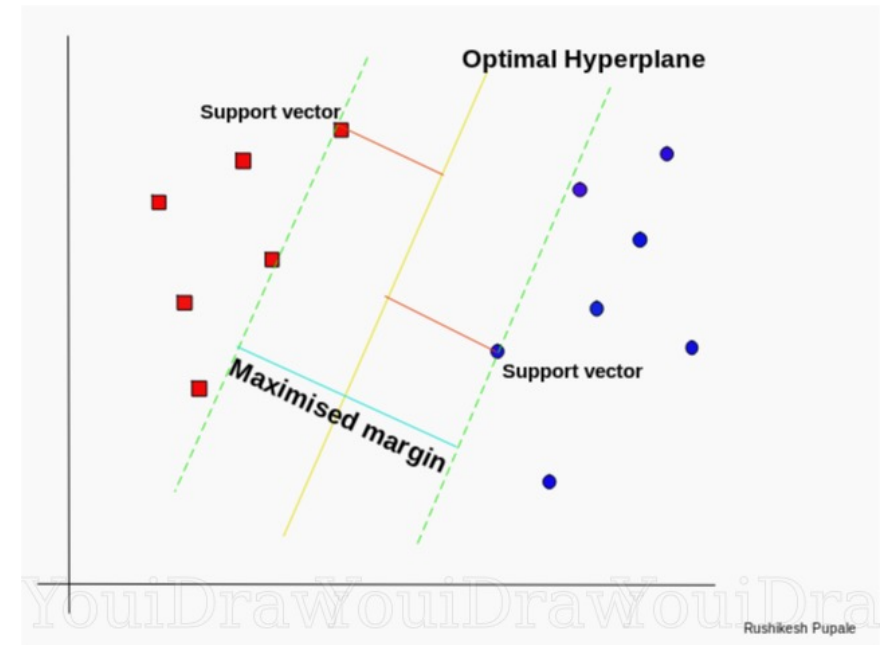
Ex: Support Vector Machine

- ⦿ SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible
- ⦿ But which line is better?



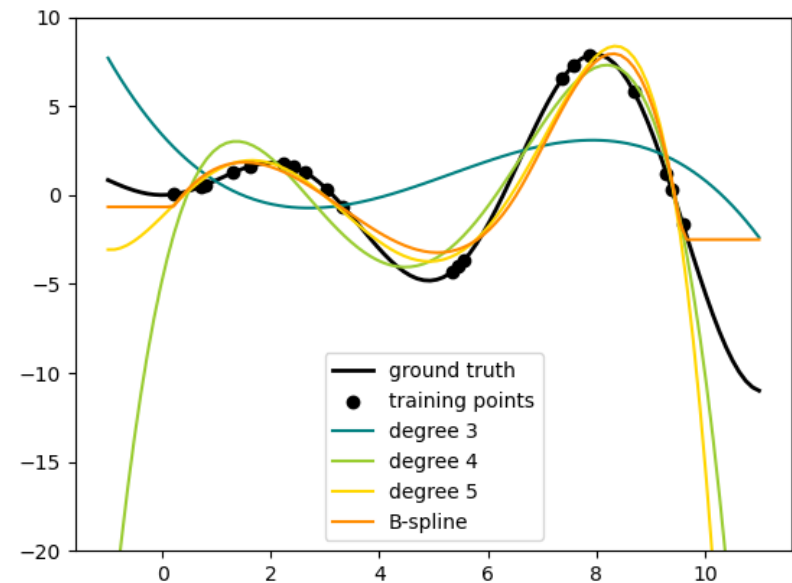
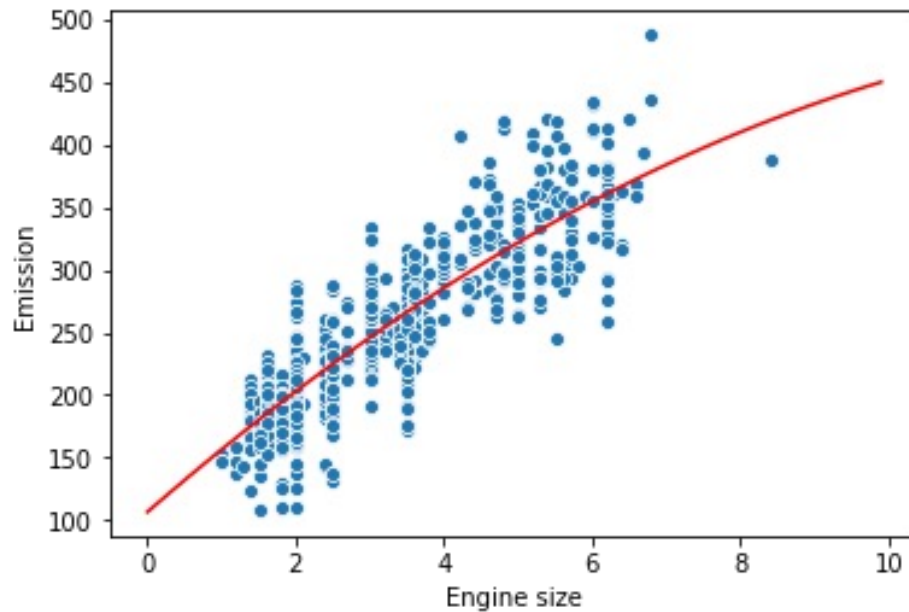
SVM principle

- ⦿ According to the SVM algorithm we find the points closest to the line from both the classes
- ⦿ These points are called support vectors
- ⦿ Now, we compute the distance between the line and the support vectors
- ⦿ This distance is called the margin
- ⦿ SVM tries to maximize the margin
- ⦿ The hyperplane for which the margin is maximum is the optimal hyperplane.



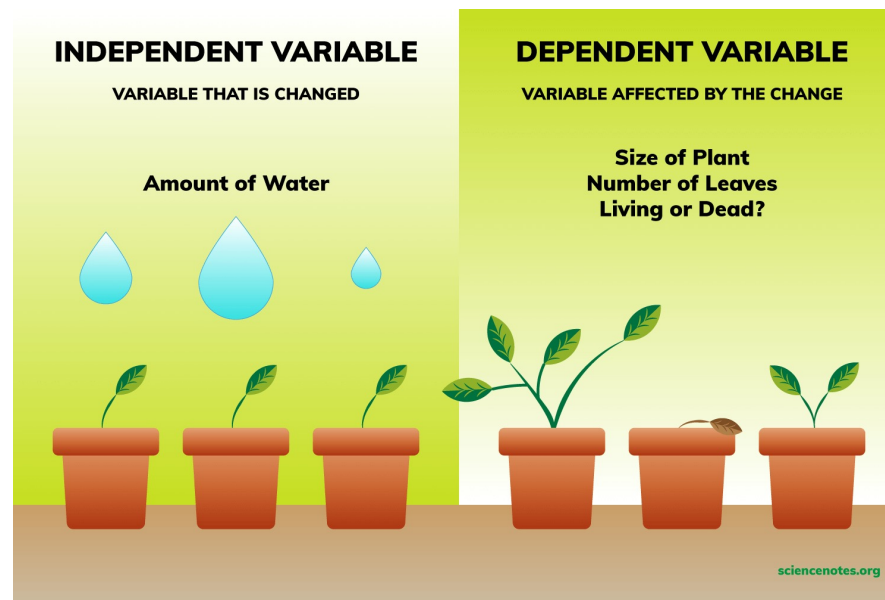
Regression in real-world problem?

- Here again, the choice of the regression method has obviously high impact: linear/nonlinear, simple/multiple, error model, ...



More about regression

- ⦿ **Dependent variables:** the main event or factor to understand or predict. Also known as *explanatory variable*.
- ⦿ **Independent variables:** the events or factors suspected to have an impact on the dependent variable. Also known as *response variable*.



Types of Regression

- **Simple regression:** single independent variable x for a single dependent variable Y .

x : number of cricket chirps

Y : temperature

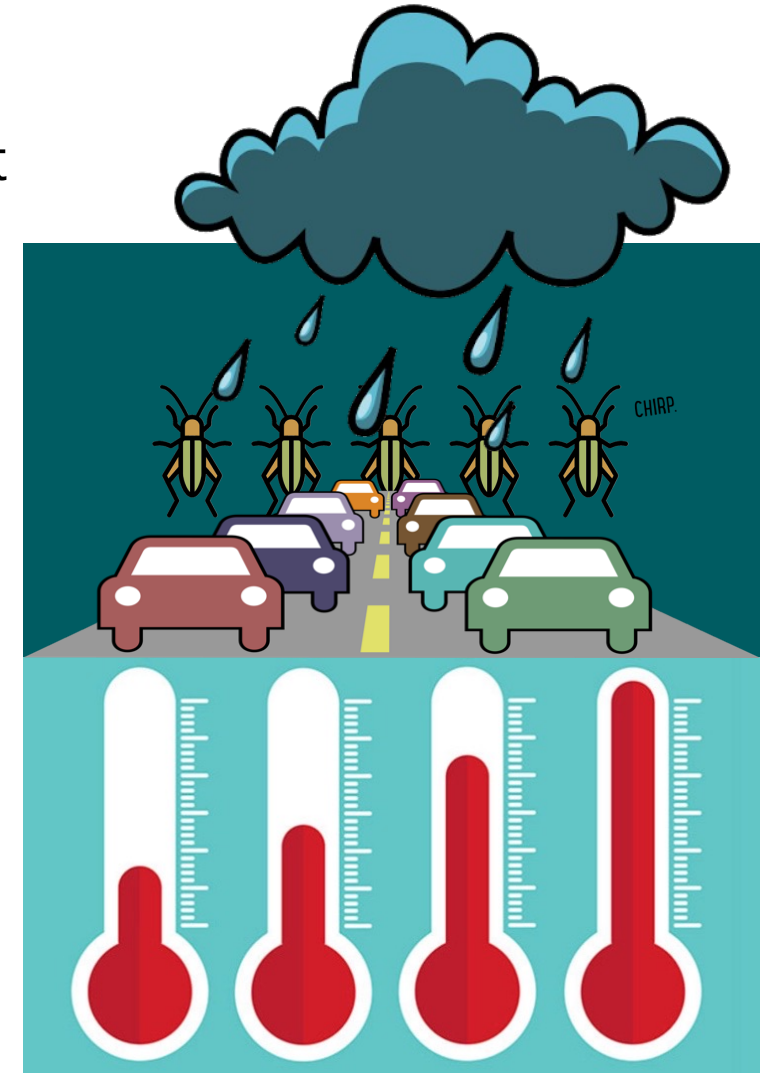
- **Multivariable regression:** multiple independent variables, x_1, x_2, x_3 , for a dependent variable Y .

x_1 : number of cricket chirps

x_2 : rainfall

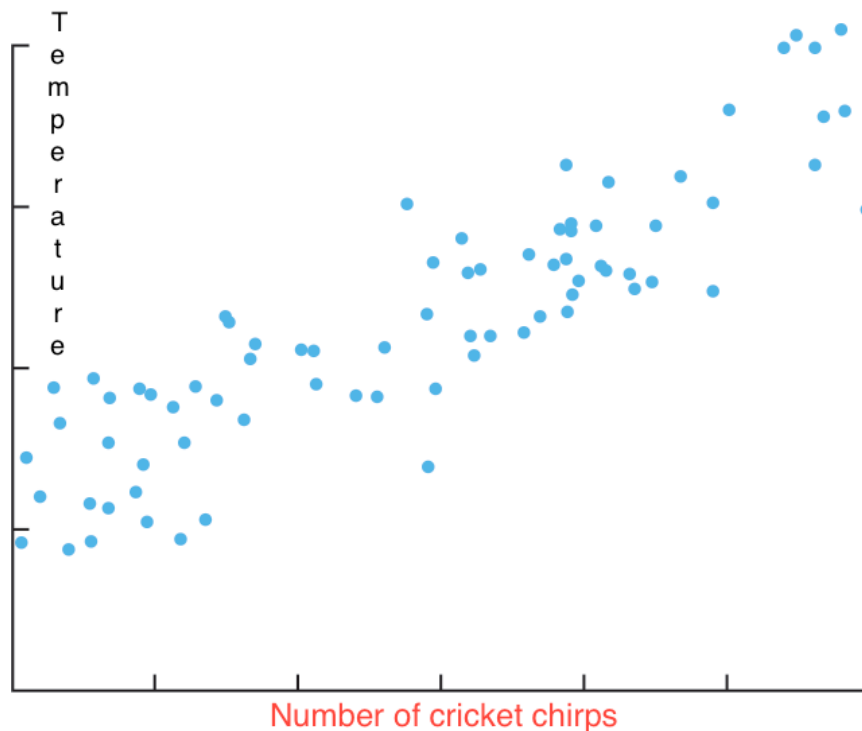
x_3 : automobile traffic

Y : temperature



Scatter Plot of datasets

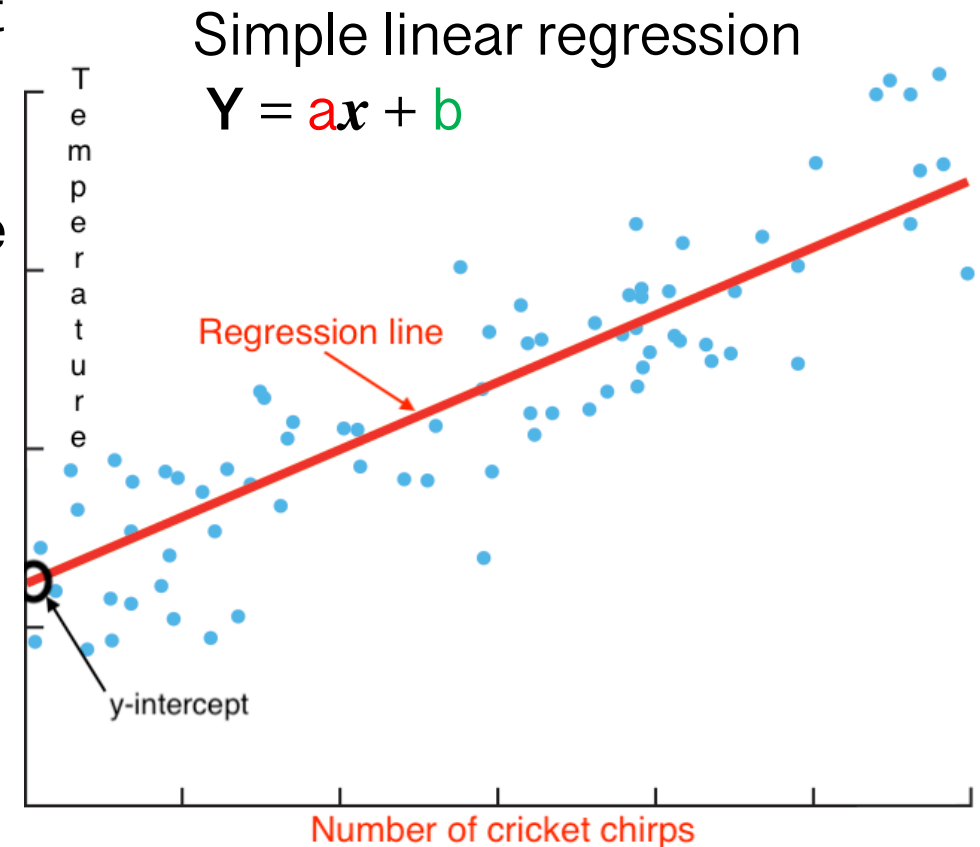
- ⦿ Data gathering on the variables in question
- ⦿ The vertical scale represents one set of measurements and the horizontal scale the other



- ⦿ Agriculture
 - ⦿ Soil moisture for optimized irrigation
- ⦿ Environment
 - ⦿ Pollutant concentration for public safety and alarming system
- ⦿ Electricity
 - ⦿ Power demands for optimized production
- ⦿ ...

Linear Regression

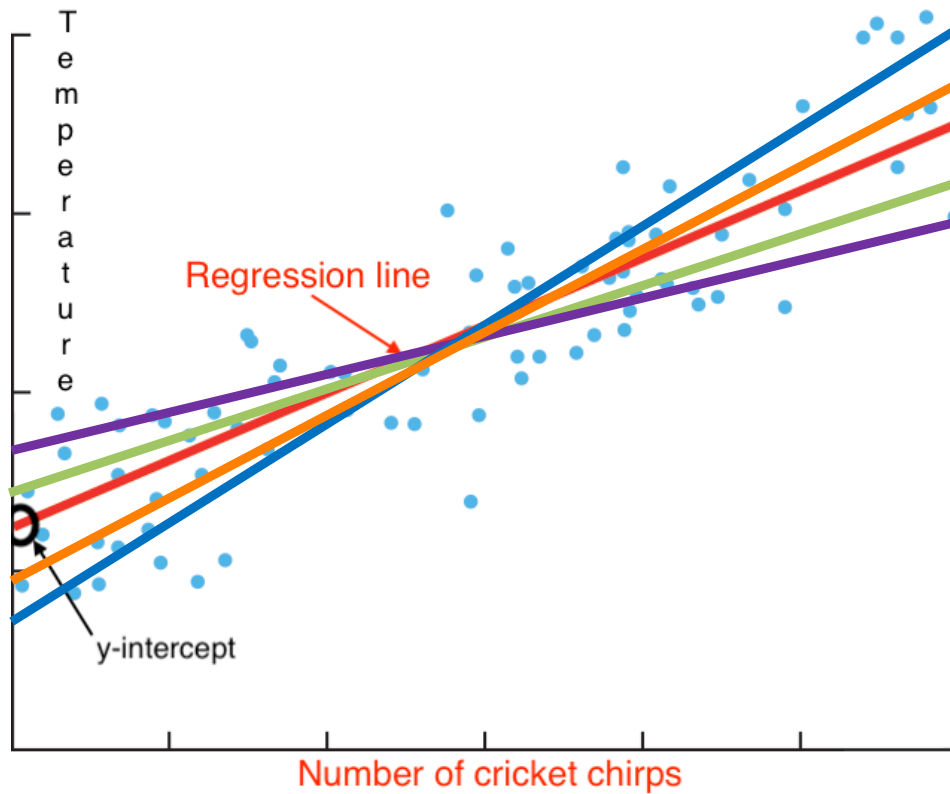
- ⦿ A linear relationship to predict the (average) numerical value of Y for a given value of x using a straight line, called the *regression line*.
- ⦿ Knowing the *slope* and the *y*-intercept the objective is to predict the average Y from x .
- ⦿ Real world problem can be multivariate
- ⦿ Multiple linear regression



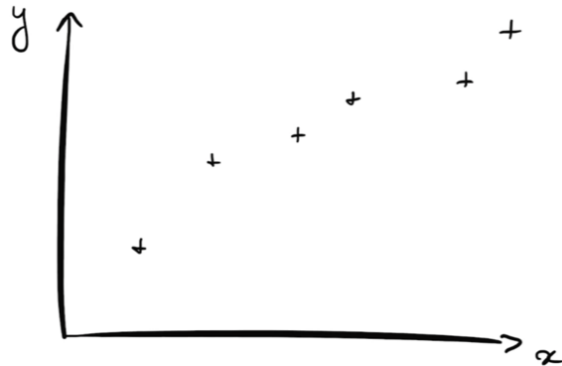
$$Y = a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_ix_i + b$$

Then what?

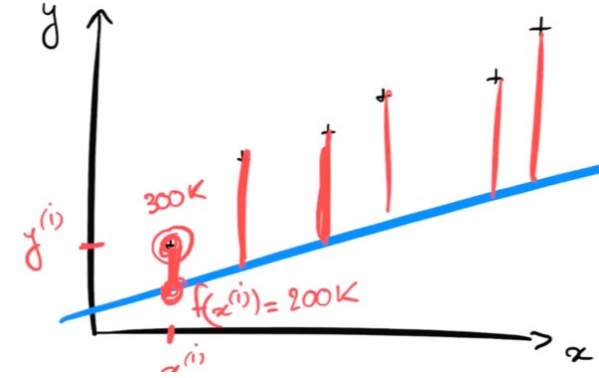
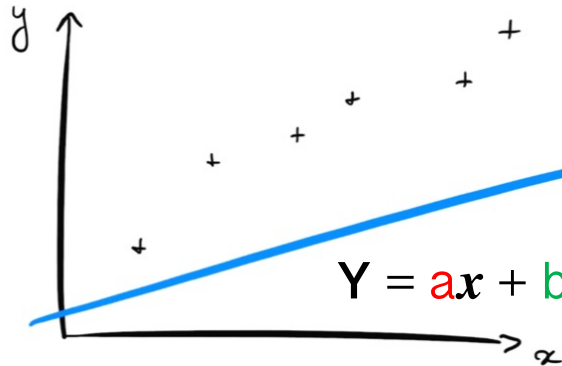
- ⦿ What is the best regression line?
- ⦿ How can we find it?



The need for a cost function

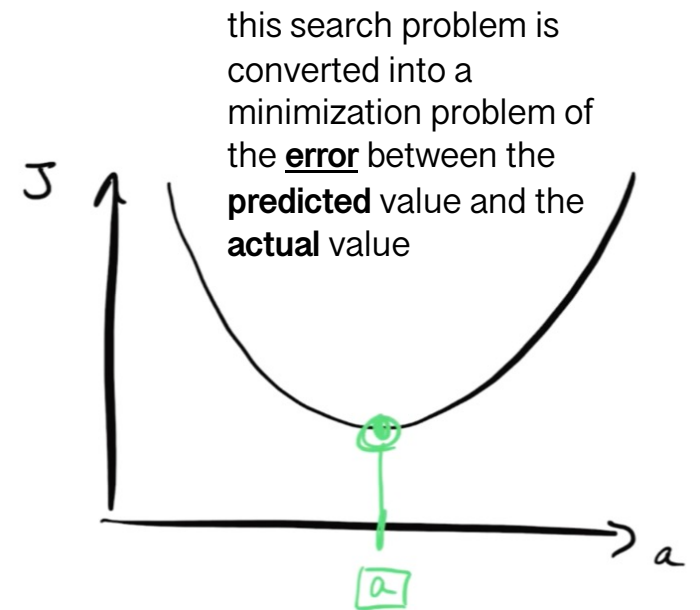
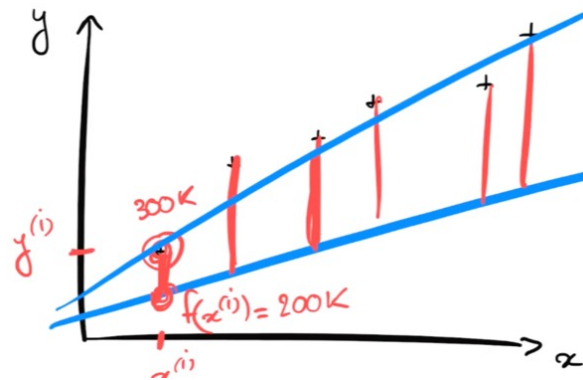


1. Dataset: (x, y) $m = 6, n = 1$ $x^{(i)}$



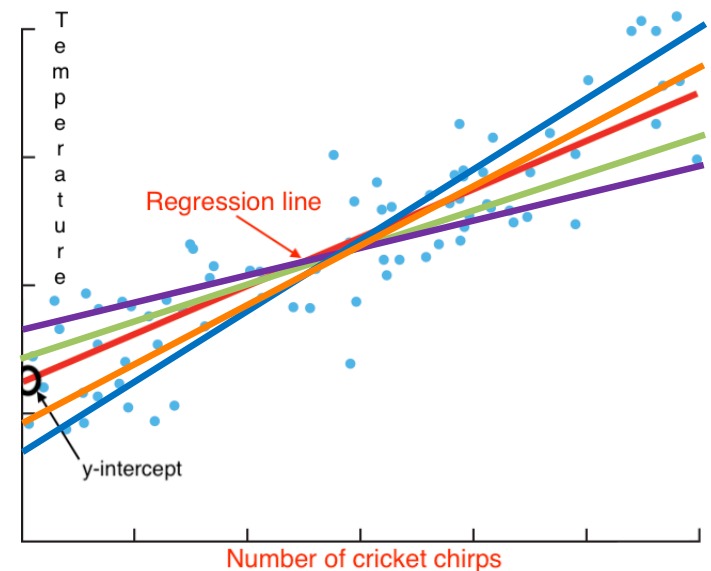
$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$

A **cost function** will help to figure out the best possible values for **a** and **b** which would provide the best *regression line* for data points
Here, Mean Squared Error (MSE)



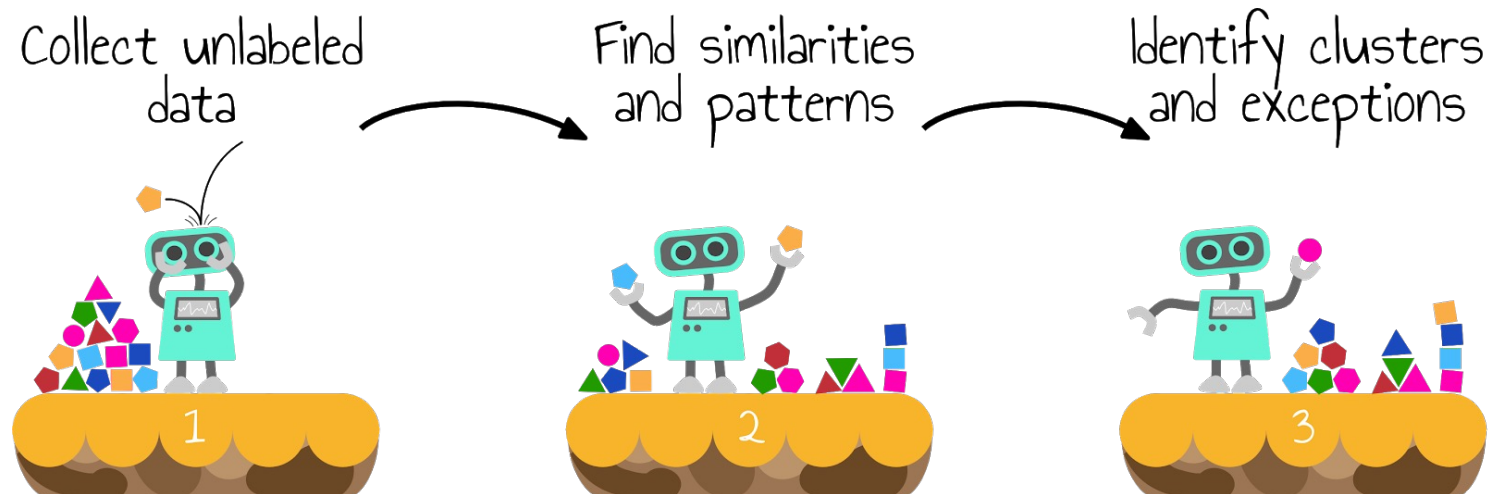
Why do we call it Machine Learning?

- ⦿ The number of data points can be very large!
- ⦿ The number of possible regression lines can be very large!
- ⦿ Trying in a brute force approach can take years, even with state-of-the-art supercomputers!
- ⦿ Exact methods also need very complex & long operations to be conducted!
- ⦿ It is called AI/ML because we "tell" the machine to efficiently find the best solution
- ⦿ "tell"=implement "intelligent" algorithms!



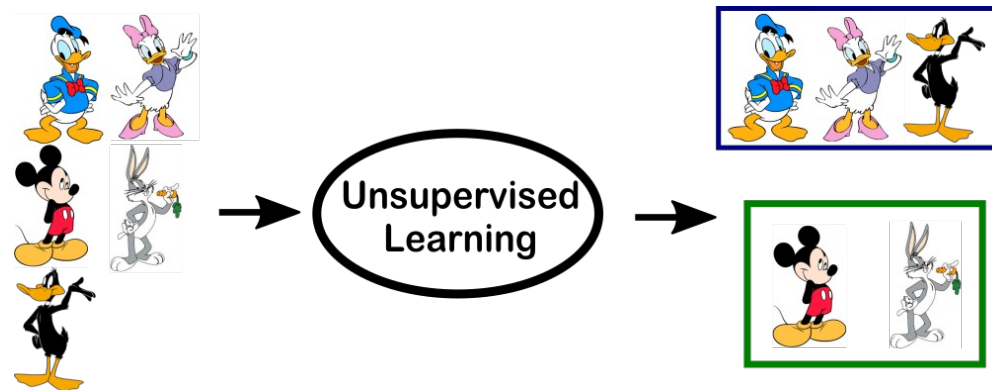
Unsupervised Learning – clustering

- ⦿ Mainly used for **clustering** to learn patterns from untagged data
- ⦿ Find natural groups in the feature space of input data
- ⦿ No single best method for all datasets
- ⦿ Models are computationally complex because they need a large training set to produce intended outcomes
- ⦿ Still require some human intervention for validating output



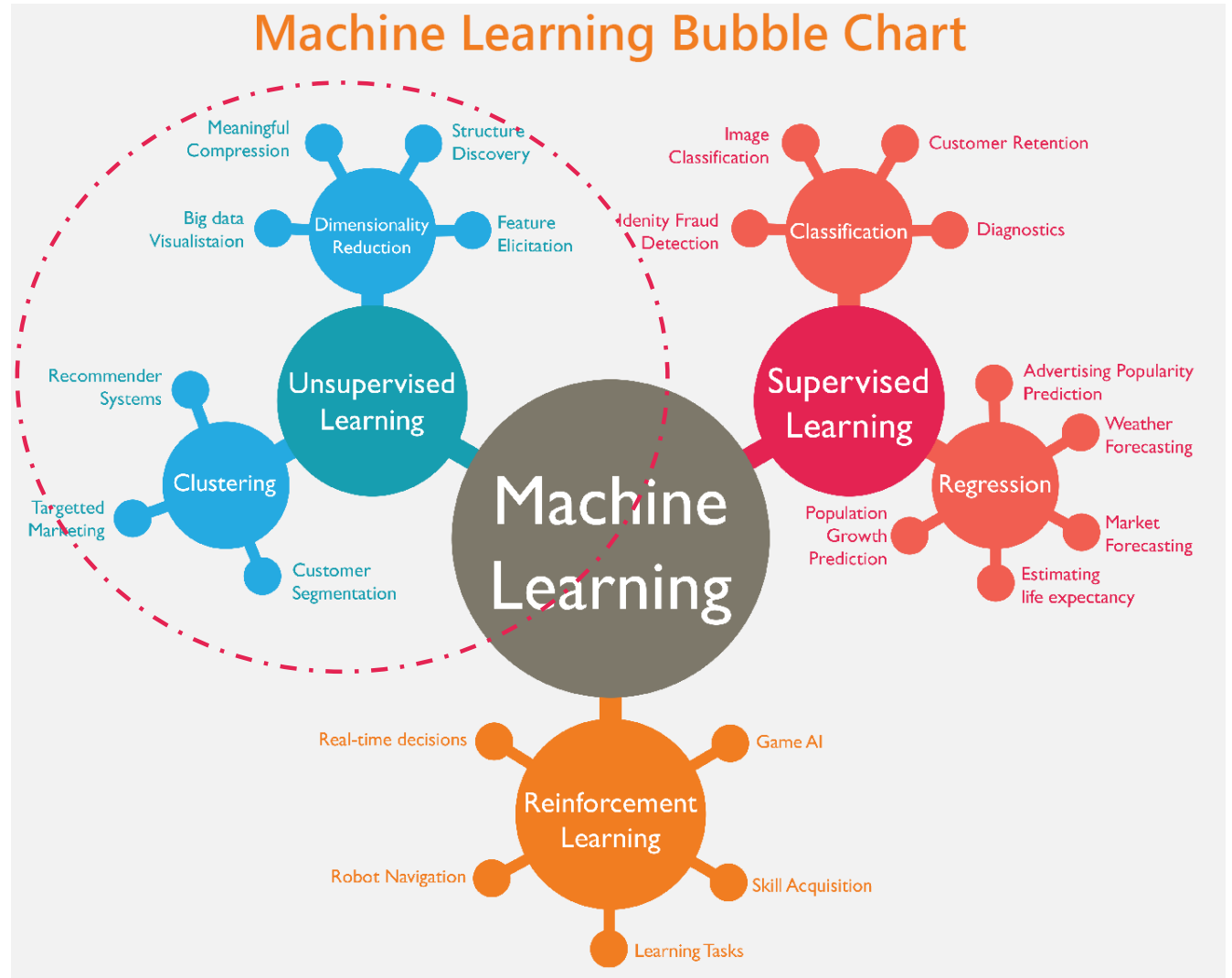
Ducks example

- ⦿ In the below example, some cartoon characters are passed to the ML model where some of them are ducks
- ⦿ No data label provided
- ⦿ ML model is able to separate the characters into **Duck** and **No Duck** by looking at the type of data and models in the underlying data structure



Clustering algorithms

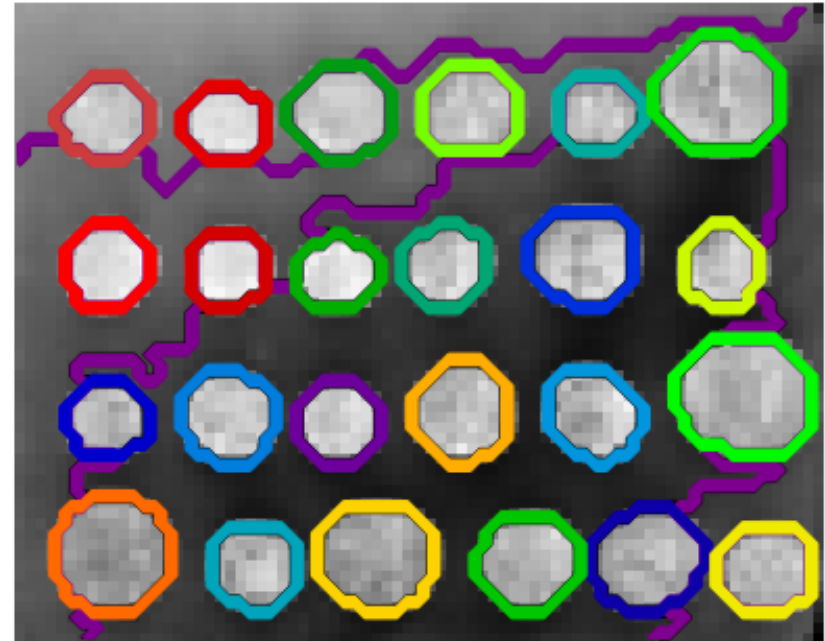
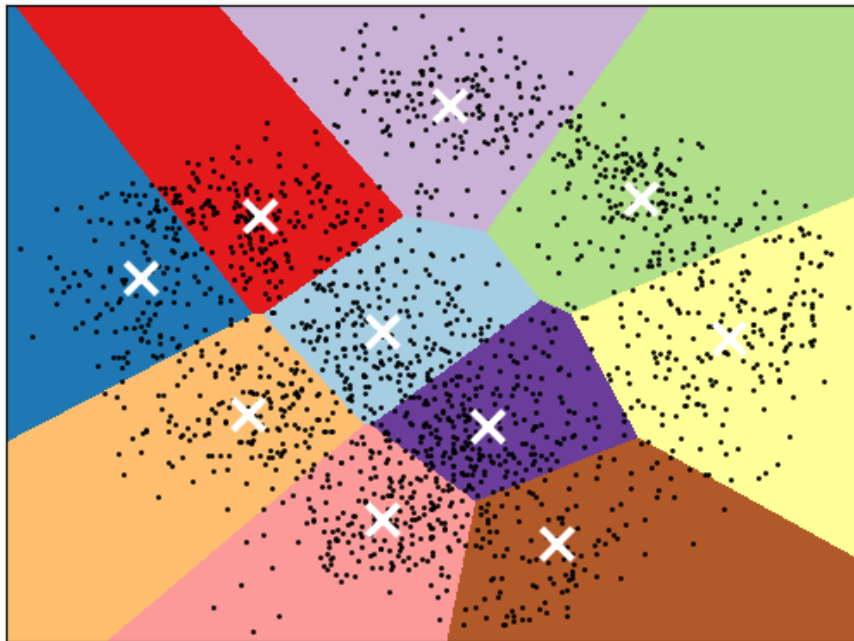
- Clustering
 - Affinity Propagation
 - Agglomerative Clustering
 - BIRCH
 - DBSCAN
 - K-Means
 - Mini-Batch K-Means
 - Mean Shift
 - OPTICS
 - Spectral Clustering
 - Mixture of Gaussians
 - ...



Example of advanced clustering

- There is no best clustering algorithm, and no easy way to find the best algorithm for your data without using controlled experiments

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross



Trying some clustering algorithms

Assignment to do in lab

- ① 10 Clustering Algorithms With Python
- ① <https://machinelearningmastery.com/clustering-algorithms-with-python>

10 Clustering Algorithms With Python

by Jason Brownlee on [April 6, 2020](#) in [Python Machine Learning](#)

[Tweet](#) [Tweet](#) [Share](#) [Share](#)

Last Updated on August 20, 2020

Clustering or **cluster analysis** is an unsupervised learning problem.

It is often used as a data analysis technique for discovering interesting patterns in data, such as groups of customers based on their behavior.

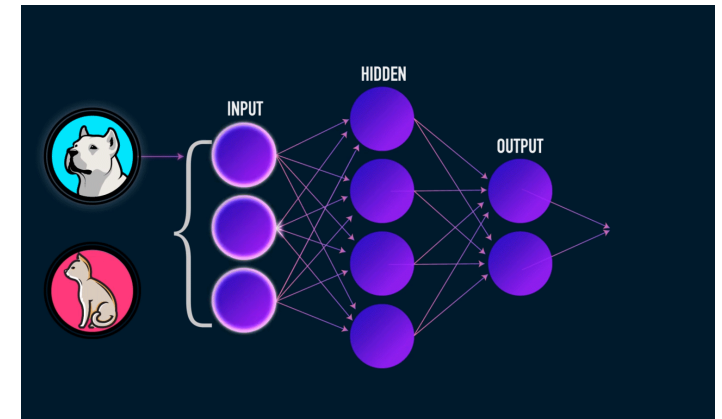
There are many clustering algorithms to choose from and no single best clustering algorithm for all cases. Instead, it is a good idea to explore a range of clustering algorithms and different configurations for each algorithm.

In this tutorial, you will discover how to fit and use top clustering algorithms in python.

- ① You can use Google Colab to test some clustering algorithms

What about image recognition?

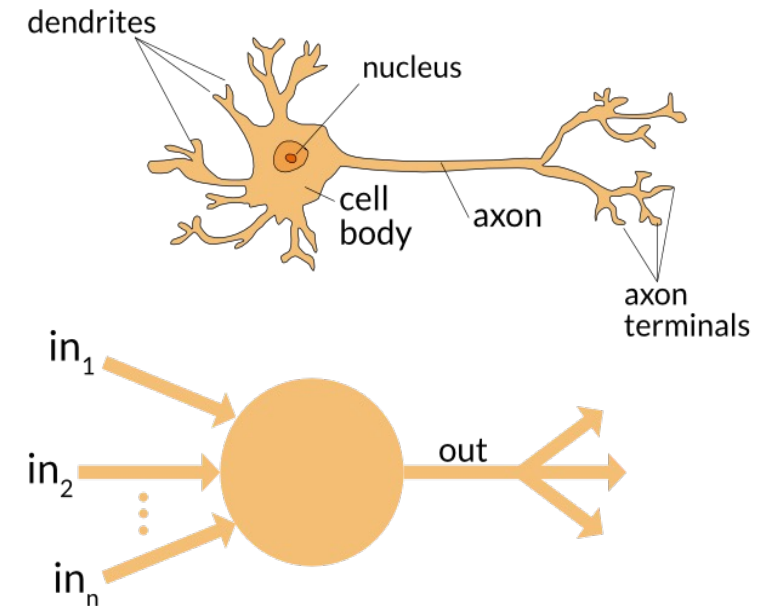
- Image recognition (or image classification) is probably the most known achievement of modern AI seen by the general public thanks to Google, Facebook, ...
- Image recognition is one of the tasks in which Deep Neural Networks (DNNs) excel
- DNN are at the foundation of so-called Deep Learning (DL) techniques which is a subset of ML techniques
- Most of image recognition approaches – thus DL – belong to the supervised ML category



TowardsDataScience

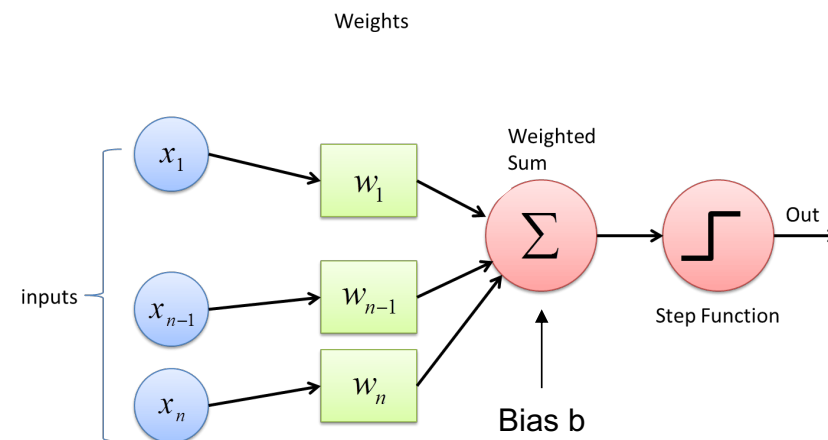
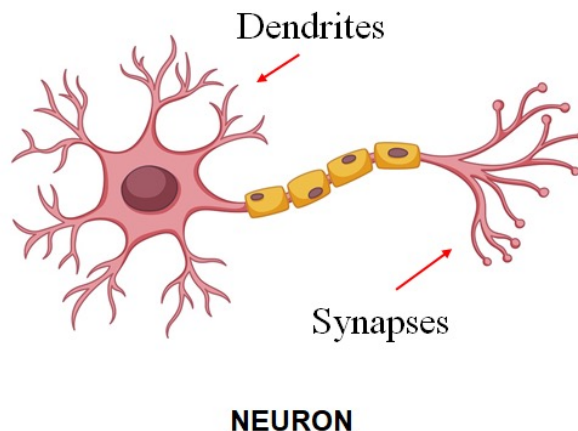
More like humans? Towards DL!

- ⦿ In early AI ages, researchers wanted to mimic the human brain seen as a network of neurons
- ⦿ **Perceptron**: Mathematical representation of a biological neuron
- ⦿ First implementation by Frank Rosenblatt in 1958
- ⦿ Rosenblatt's perceptron is activated when there is sufficient stimuli or input
- ⦿ Neurons have been found to perform a similar process, in which experience strengthens or weakens dendrites' connections



How does a Perceptron work?

- Single Layer Perceptron (SLP) receives the value of the attributes of an input, just as dendrites do in a neuron
- Each attribute has a **weight** w that measures its *contribution* to the final result, which is the sum of the multiplications of inputs of each attribute by its corresponding weight
- The final output depends on the activation function: here, if the sum is >0 Perceptron returns a value of 1, otherwise it yields 0



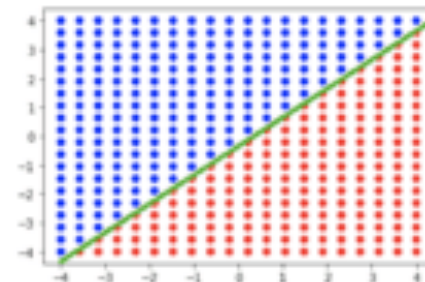
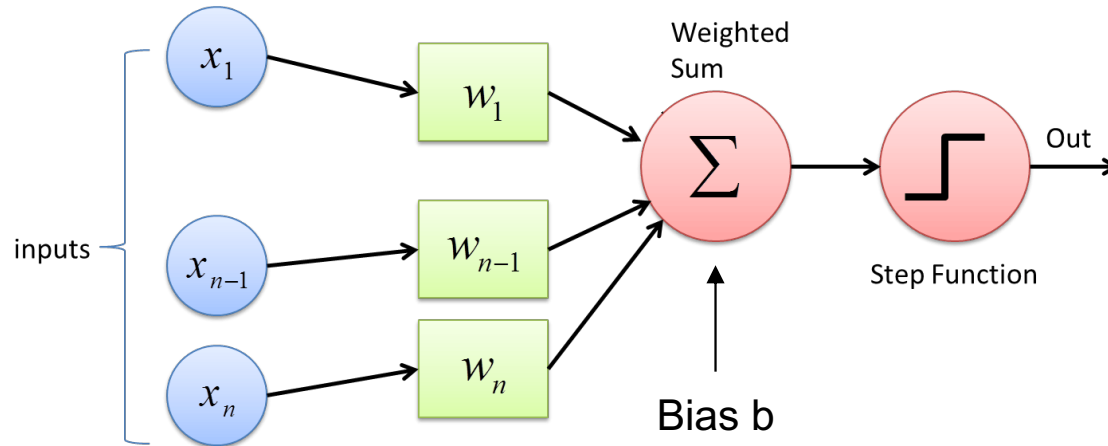
What can an SLP do?

Weights

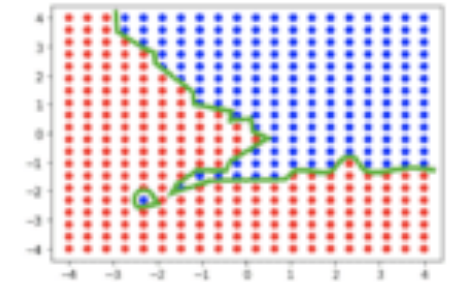
$$\hat{y} = \Theta(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$

$$= \Theta(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\text{where } \Theta(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

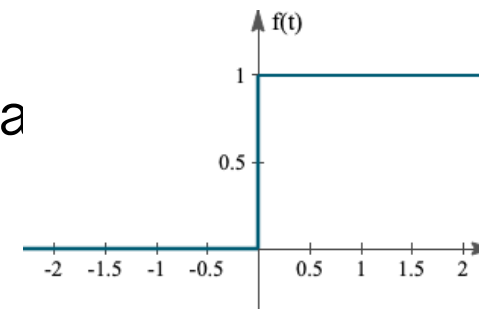


Linear boundary



Non-linear boundary

- SLP is the simplest type of artificial neural networks and can only classify **linearly separable cases** with a binary target
- Activation functions are mathematical equations that determine the output of a neural network
- Here, the **Heaviside step function** Θ



Linearly vs nonlinearly separable

- Most real world problems are nonlinearly separable!

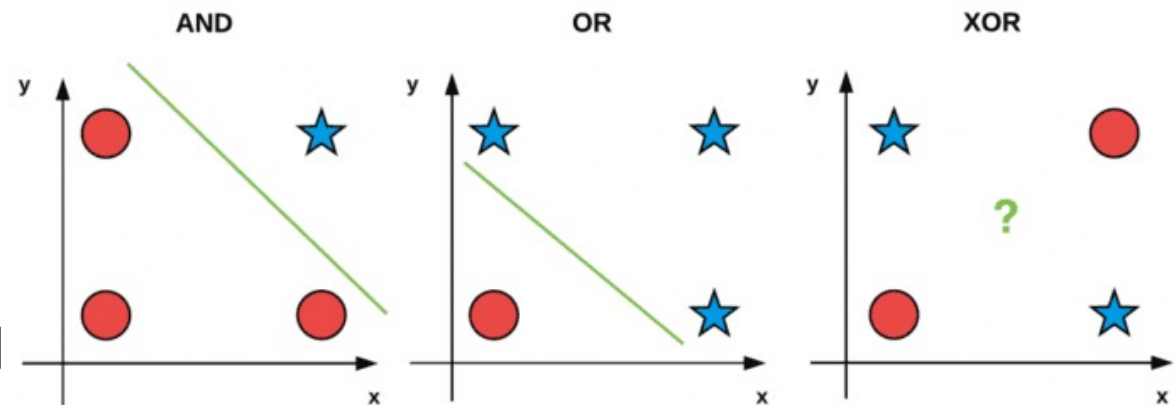
- OK, but can you find a clear example?

- Let's try to better understand nonlinearly separable cases on a very simple example

- Take the AND, OR and XOR logic gates

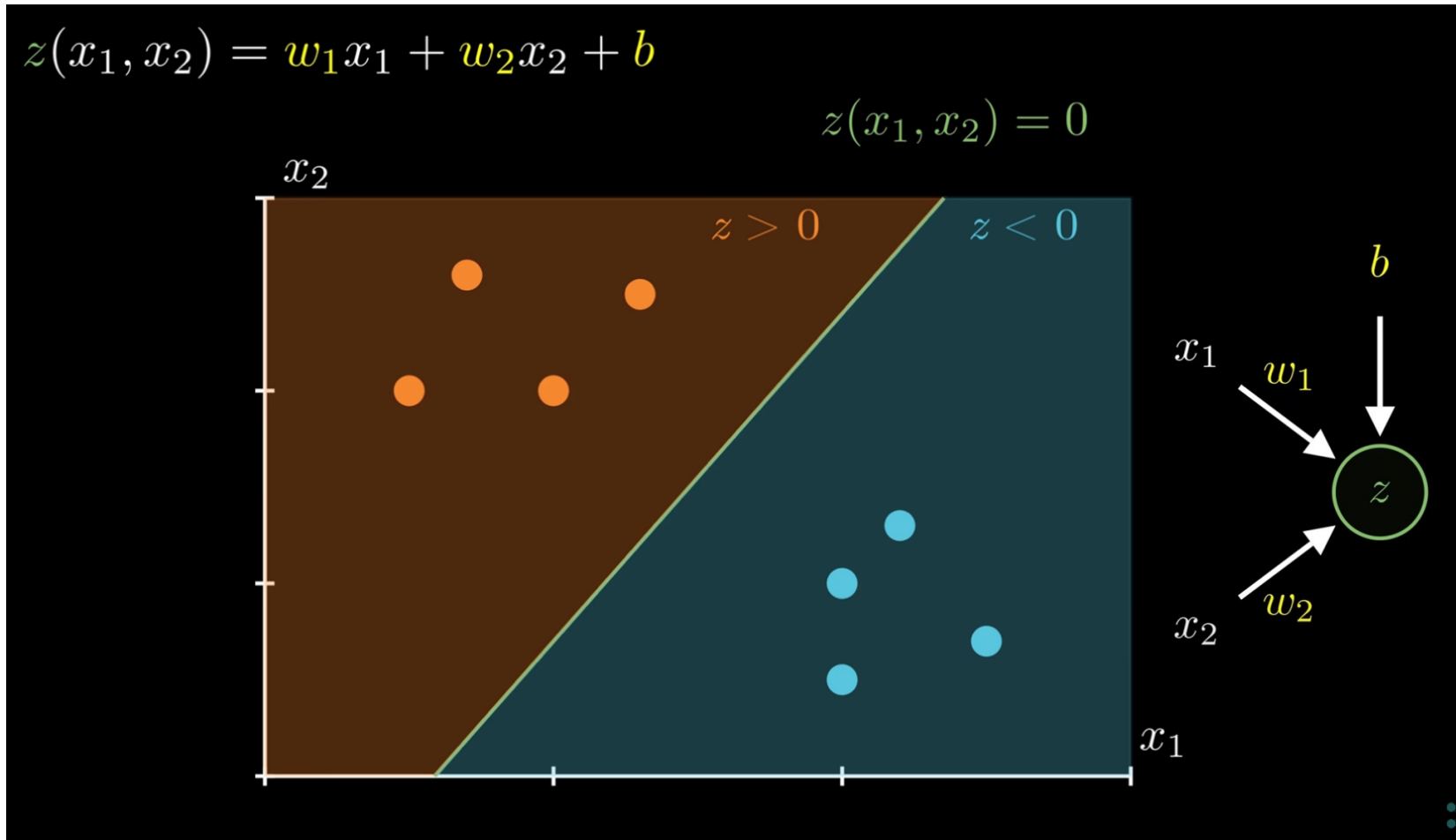
- AND and OR are **linearly separable**

- XOR is **nonlinearly separable!**

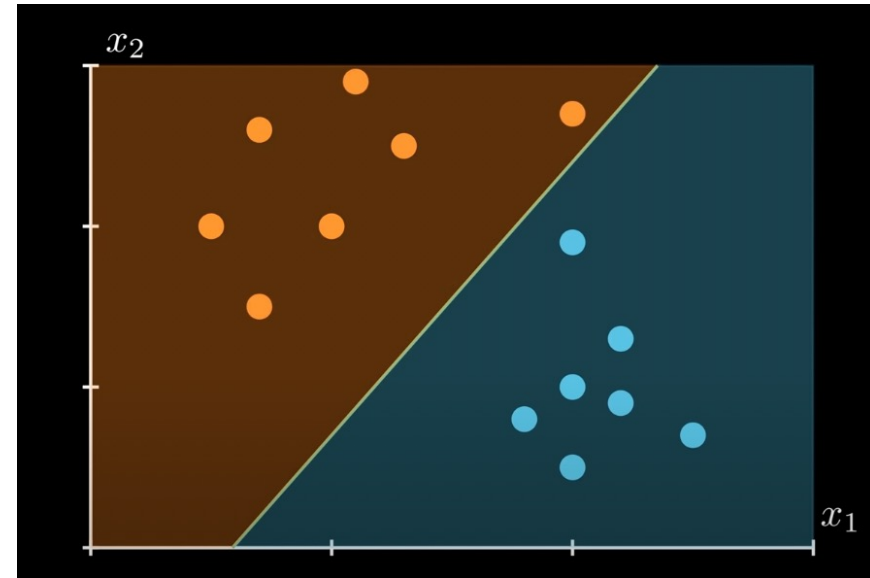
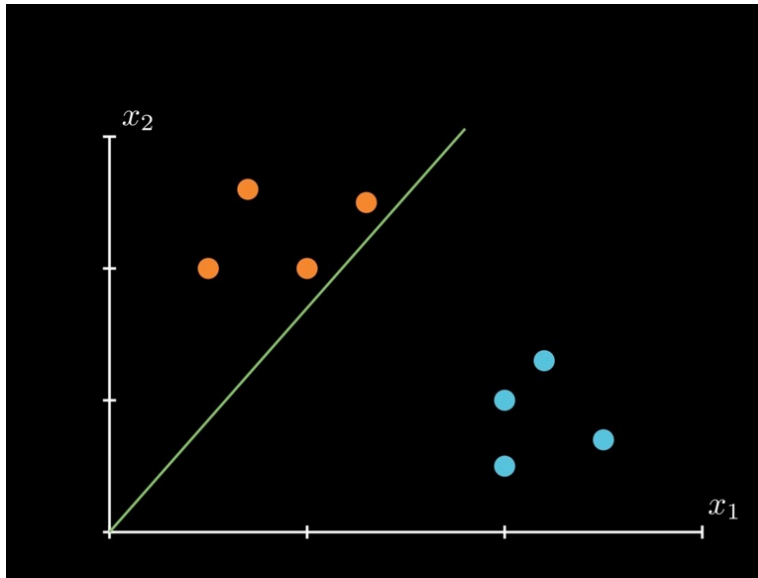


x_0	x_1	$x_0 \& x_1$	x_0	x_1	$x_0 x_1$	x_0	x_1	$x_0 \wedge x_1$
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

The perceptron classifier



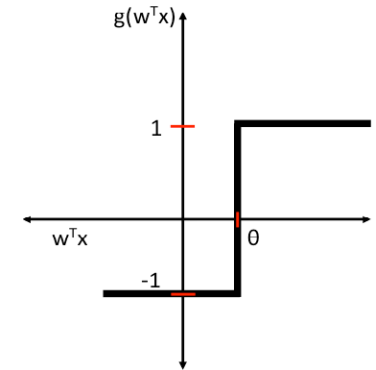
The role of the bias b



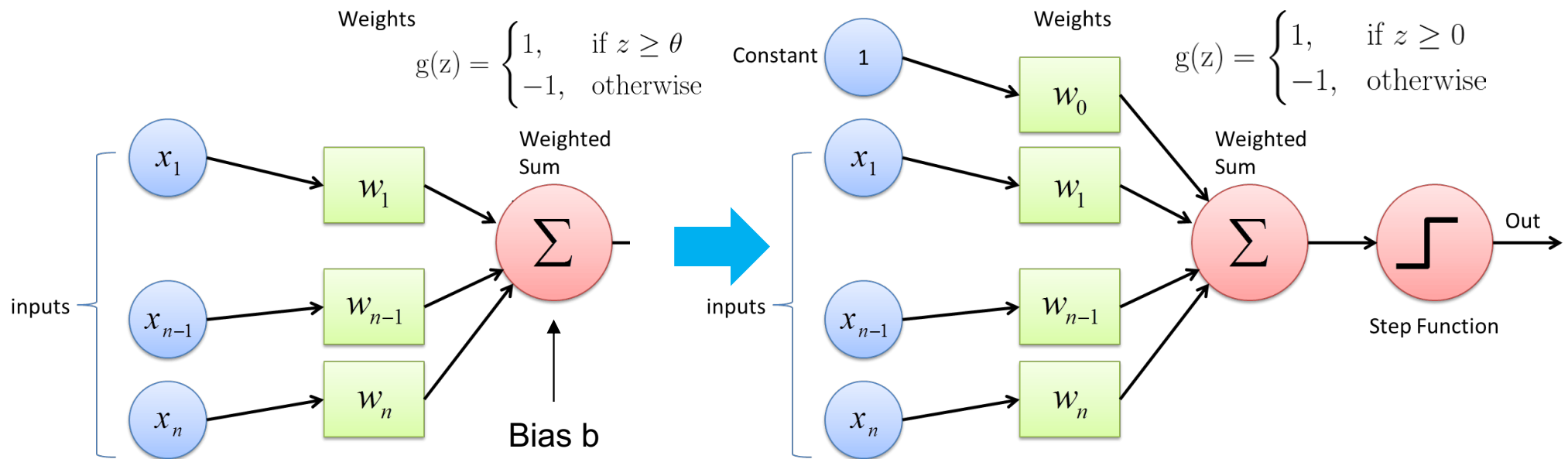
- ⦿ When $b=0$, the separation line (decision boundary) goes from the origin
- ⦿ With a bias b , it is possible to move the decision boundary to better divide the 2 target areas

Adding the bias as an input

- ⦿ In general, we define an activation function $g(z)$, where if $g(z)$ is greater than a defined threshold θ we classify accordingly (1 or -1 for instance)
- ⦿ To simplify the notation, we bring θ to the left side of the equation and define $w_0 = -\theta$ and $x_0 = 1$ (also known as bias)

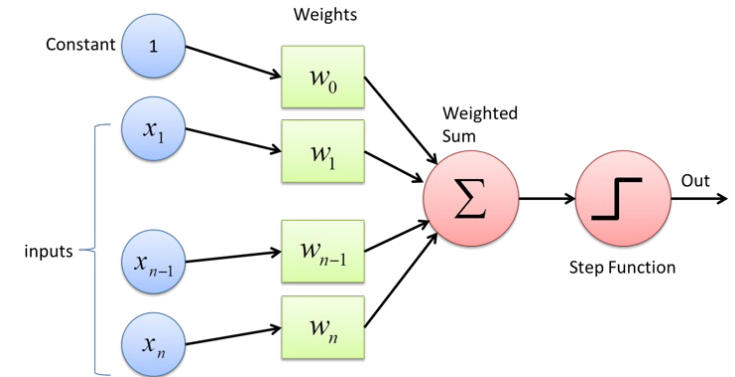


Pr. Congduc Pham
http://www.univ-pau.fr/~cpham



Dot product and matrix notation

$$z = \sum_{j=0}^m x_j w_j = w^T x$$



The dot product of two vectors $\mathbf{a} = [a_1, a_2, \dots, a_n]$ and $\mathbf{b} = [b_1, b_2, \dots, b_n]$,

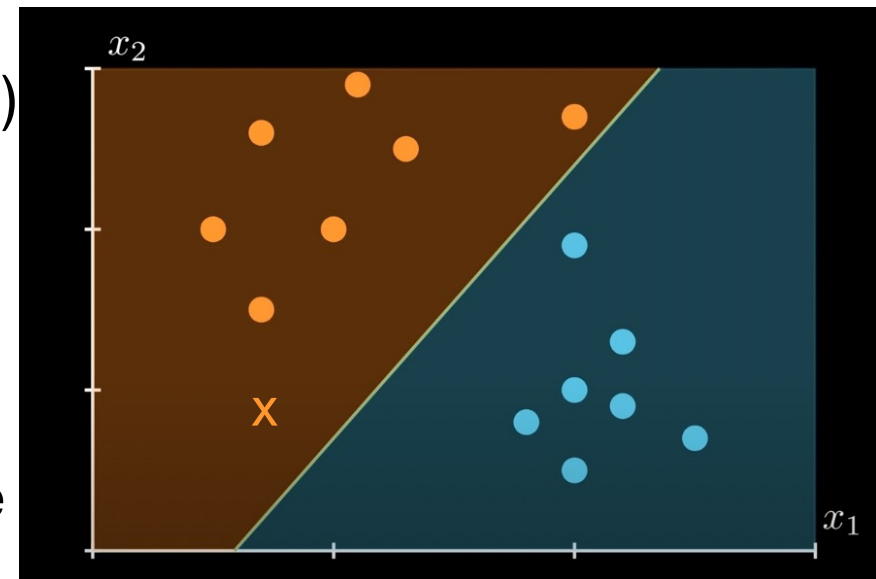
$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

- ⊙ If vectors are identified with column vectors, the dot product can also be written as a matrix product

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}, \quad [1 \quad 3 \quad -5] \begin{bmatrix} 4 \\ -2 \\ -1 \end{bmatrix} = 3$$

How does it learn?

- Learning means for the perceptron to find the right w_i and b to correctly divide the training dataset into 2 clear target classes
- Remind that the training dataset should be correctly labeled in supervised learning
- x_2 could be **wheel size** and x_1 could be **vehicle length**
- Then the 2 target classes could be **truck** (e.g. $z > 0$) and **bus** (e.g. $z < 0$)
- The training dataset would consist in a list of (x_1, x_2) that have been identified either as truck or car
- After training, the decision boundary should be able to still correctly divide the new (x_1, x_2) pair



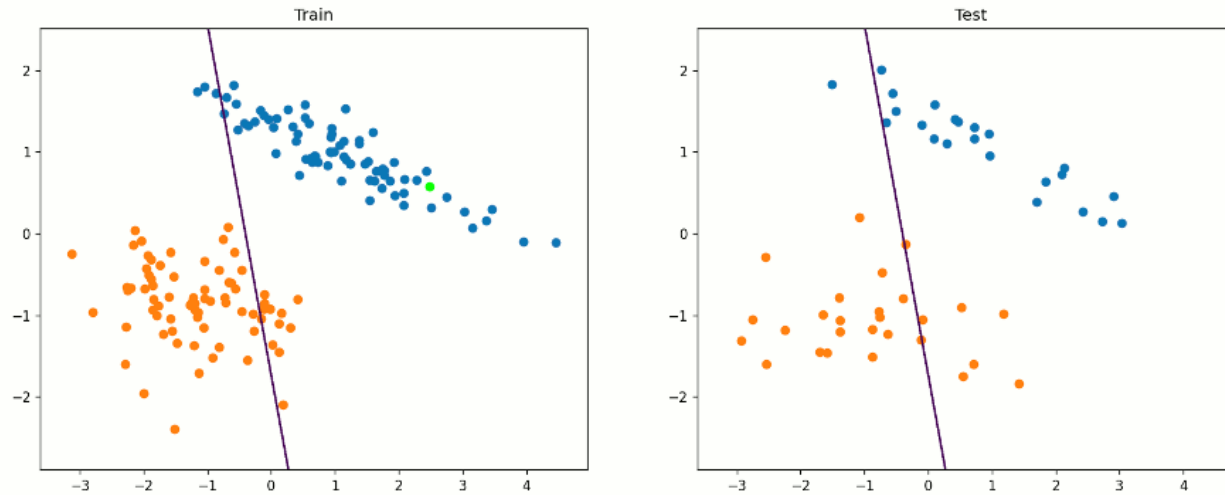
Quick overview of learning process

- ⦿ In the Perceptron Learning Rule, the predicted output is compared with the known output. If it does not match, the error is propagated backward to allow weight adjustment to happen
- ⦿ At the beginning, initialize the w_i to 0 or small random numbers
- ⦿ For each training sample x^i : calculate the output value and update the w_i
- ⦿ The output value is the class label predicted by the unit step function and the weight update can be written more formally as $w_j = w_j + \Delta w_j$
- ⦿ The value for updating the weights at each increment is calculated by the learning rule: $\Delta w_j = \eta(\text{target}^i - \text{output}^i) x_j^i$
- ⦿ where η is the learning rate (a constant between 0.0 and 1.0), target is the true class label, and the output is the predicted class label
- ⦿ In textual form: $\text{weight} = \text{weight} + \text{learning_rate} * \text{error} * \text{input}$
- ⦿ It can be time consuming and we will see that there are more efficient algorithms when the neural network becomes more complex

Learning in image

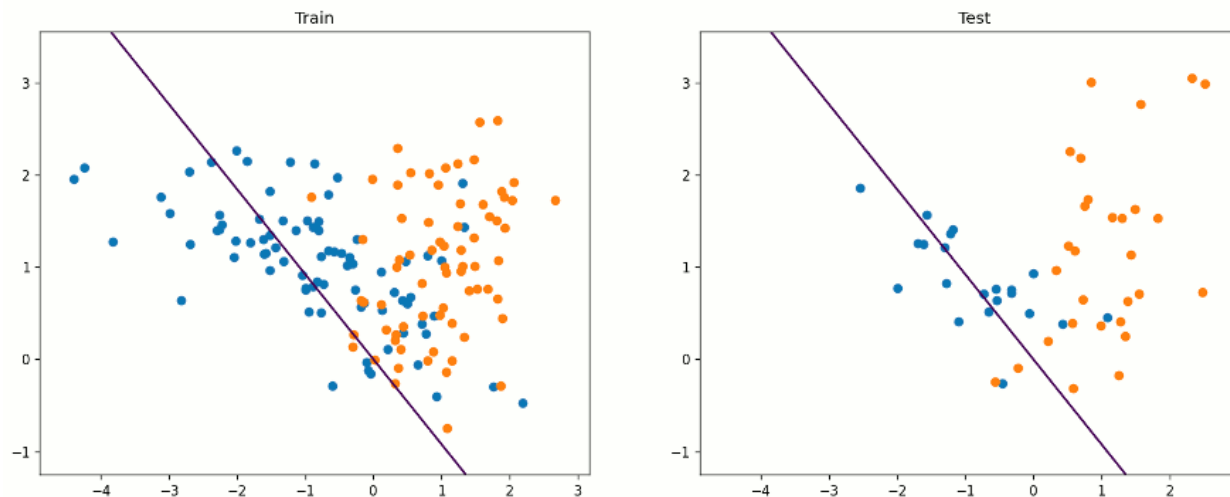
Perceptron: Explanation, Implementation and a Visual Example

Iteration: 1/2; Point: 1/150



Linearly separable

Iteration: 1/100

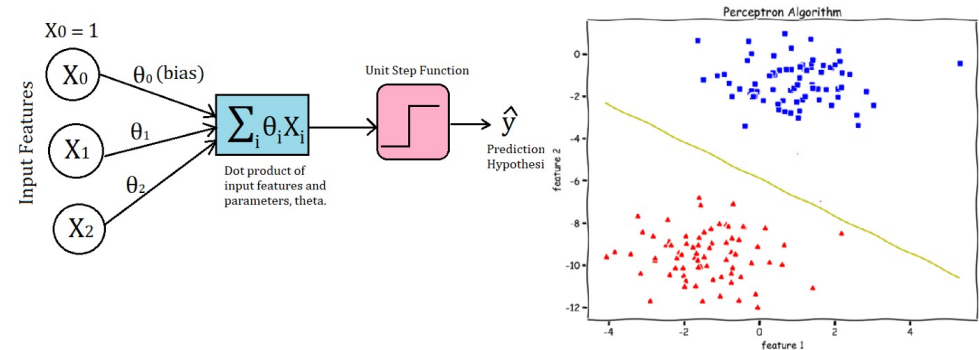


Noisy

Coding a perceptron from scratch

Assignment to do in lab

- <https://towardsdatascience.com/perceptron-algorithm-in-python-f3ac89d2e537>



- You can use Google Colab to test the code. Need to add:

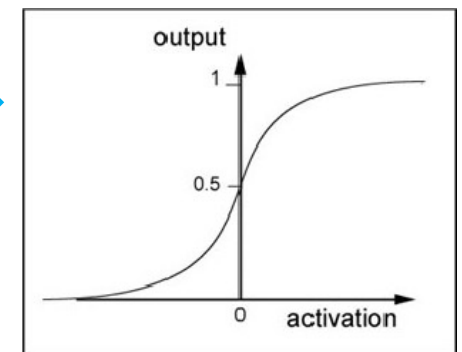
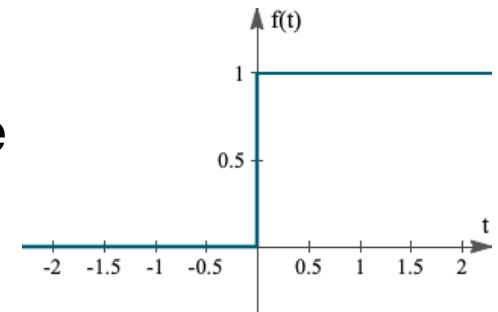
- `import matplotlib.pyplot as plt` **and** `import numpy as np`

- Other links

- <https://medium.com/@becaye-balde/perceptron-building-it-from-scratch-in-python-15716806ef64>
- <https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/>
- <https://python.plainenglish.io/building-a-perceptron-from-scratch-a-step-by-step-guide-with-python-6b8722807b2e>

Activation functions

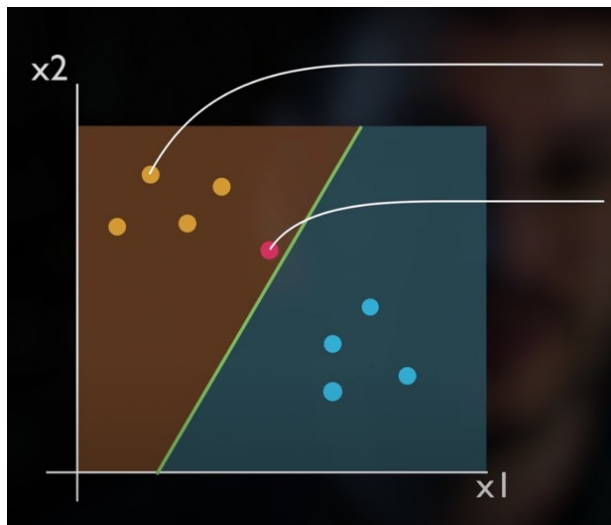
- ⊙ SLP is a feed-forward network based on a threshold transfer function which is the decision making unit
- ⊙ Previous example uses the very common Heaviside step function which produces binary output
- ⊙ The heaviside step function is typically only useful within SLP
- ⊙ There are more functions available and their interest will be described later when we will present multi-layer NN
 - ⊙ Sigmoid or Logistic (or squashing) Function
 - ⊙ Tanh or hyperbolic tangent Function
 - ⊙ ReLU or Rectified Linear Unit
 - ⊙ ...



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid Activation Function

- ⦿ Sigmoid function can be used to represent a probabilistic output
- ⦿ When classifying, some points can be considered as quite "sure" (to be **truck** or **bus** in the previous example)
- ⦿ The farther they are from the decision boundary, the more sure we can be

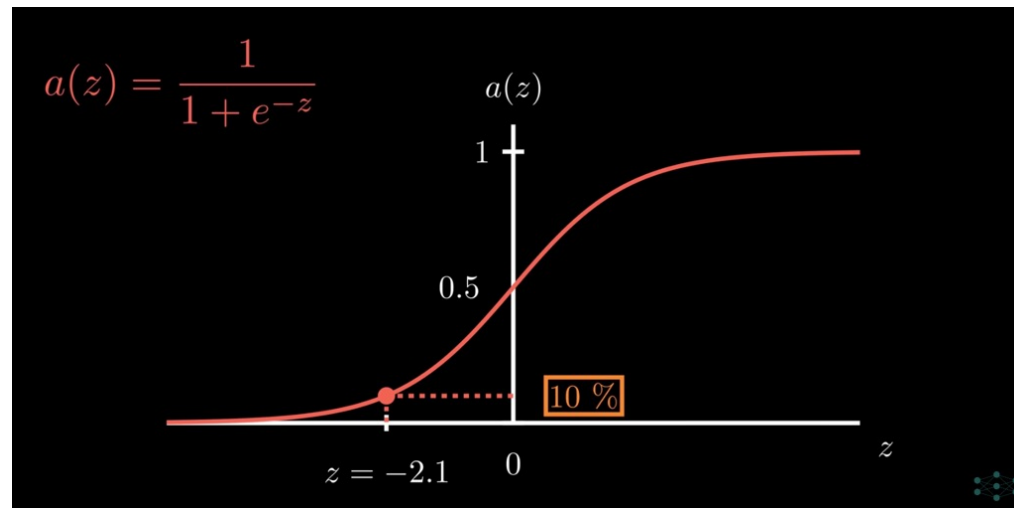
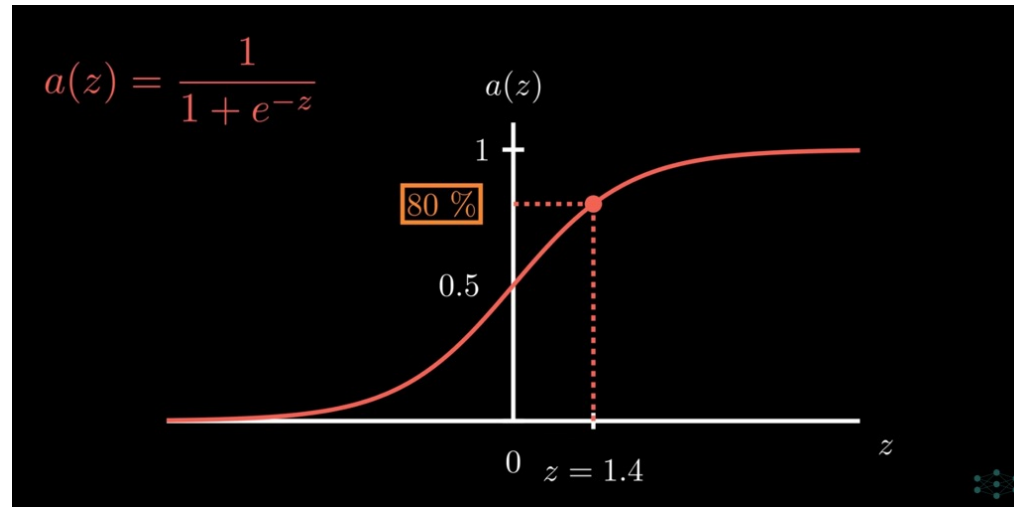


High probability

Lower probability

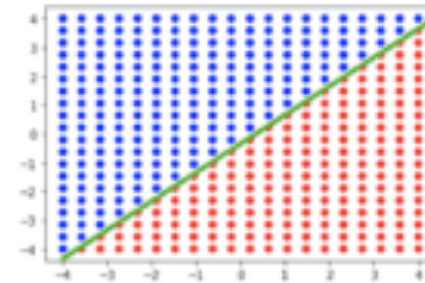


Expressing a probability...and squashing!

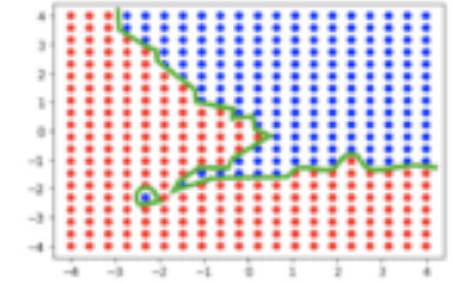


SLP limitations and how to solve them?

- SLP is the simplest type of artificial neural networks and can only classify **linearly separable cases** with a binary target



Linear boundary



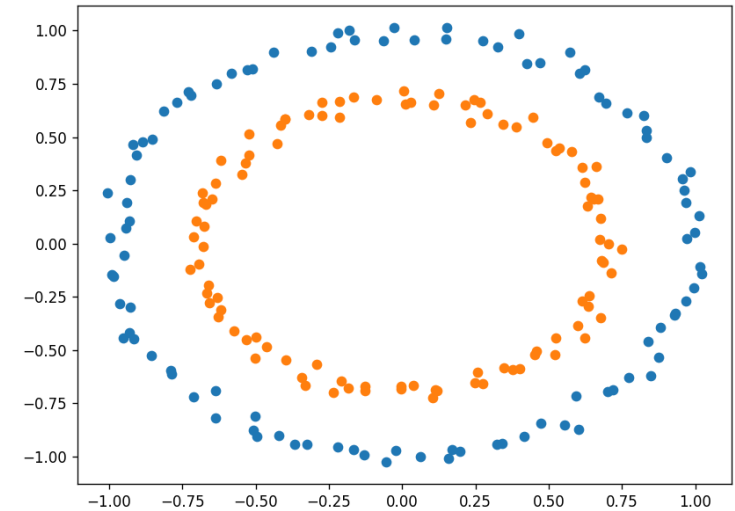
Non-linear boundary

- To solve a non-linear problem, the output should be non-linear!
- To solve a complex non-linear problem, a single non-linear output is not sufficient!
- e.g. a building is mainly made of concrete, but you probably need several competencies to make a building!



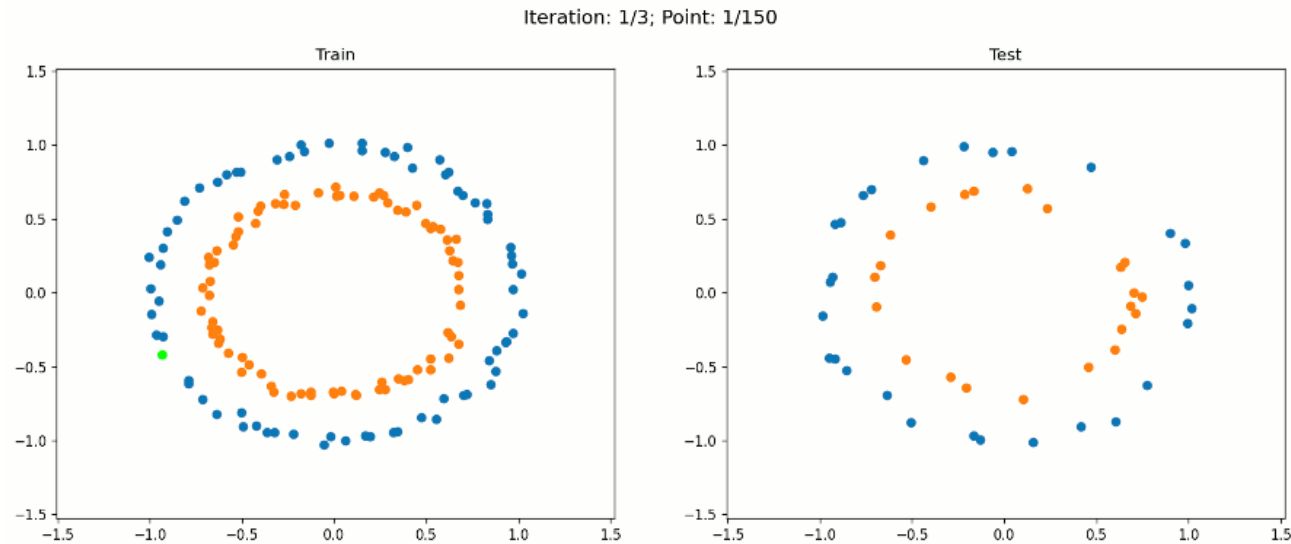
Is non-linear out of reach for SLP?

- ⦿ Consider this dataset, it is separable, but clearly not linear
- ⦿ We may think that a perceptron would not be good for this task
- ⦿ But, decision boundary is linear in terms of the weights, not necessarily in terms of inputs
- ⦿ We can augment our input vectors x so that they contain non-linear functions of the original inputs
- ⦿ For example, in addition to the original inputs x_1 and x_2 we can add the terms x_1^2 , $x_1 \cdot x_2$, and x_2^2



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

Feature augmentation, see it in image



- ⦿ The plotting uses the original inputs in order to keep it 2D. The decision boundary is still linear in the augmented feature space which is 5D now
- ⦿ But when we plot that decision boundary projected onto the original feature space it has a non-linear shape

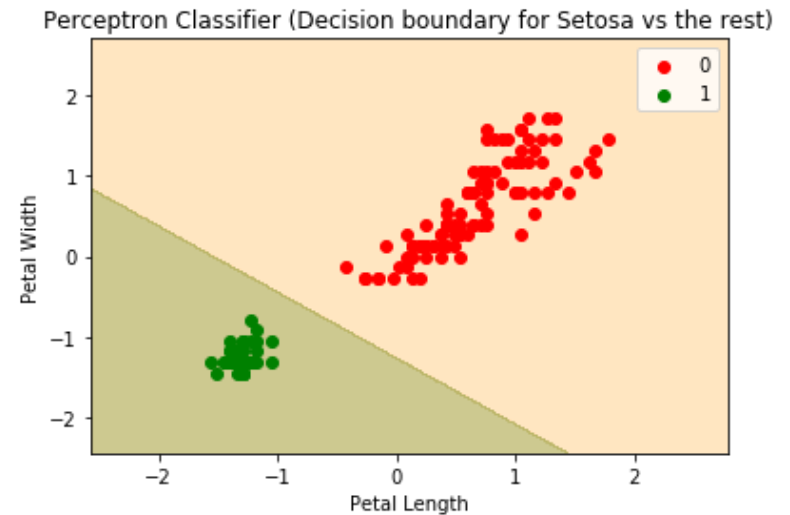
So SLP is good for non-linear as well?

- ⦿ With the previous method, our perceptron algorithm was able to correctly classify both training and testing examples without any modification of the algorithm itself – all we changed was the dataset
- ⦿ With this feature augmentation method, we are able to model very complex patterns in our data by using algorithms that were otherwise just linear
- ⦿ But, this method is not very efficient. Imagine what would happen if we had 1000 input features and we want to augment it with up to 10-degree polynomial terms!

Testing linear separability

Assignment to do in lab

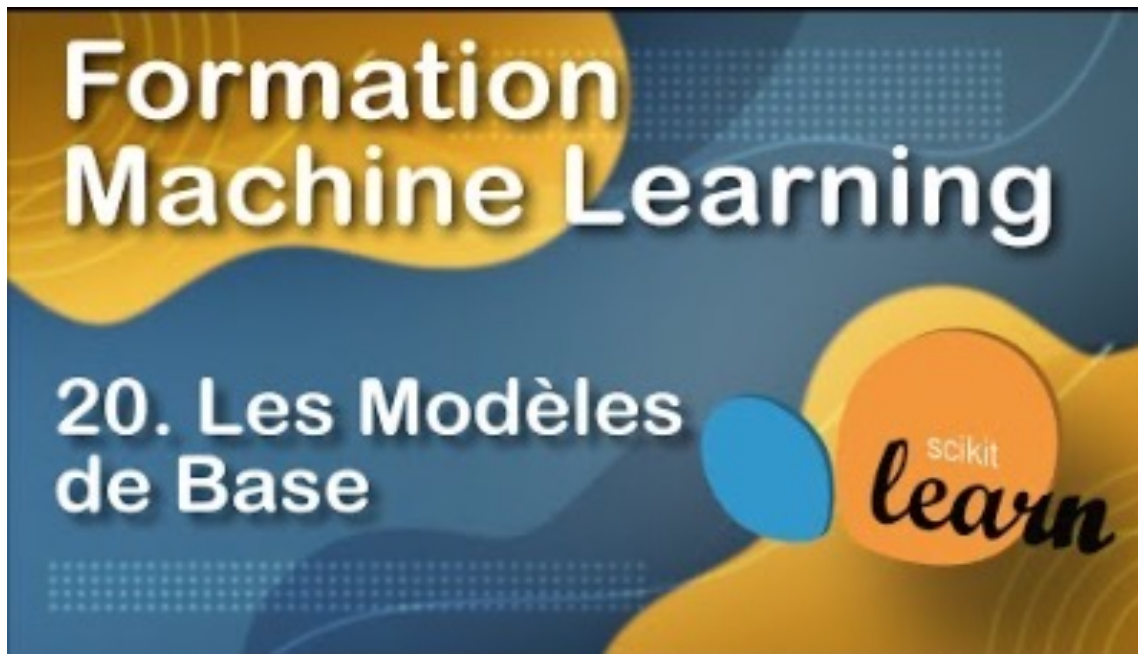
- <https://tatwan.github.io/blog/python/2017/12/31/linear-separability.html> ; <https://github.com/tatwan/Linear-Separability>



- You can use Google Colab to test the code
 - Replace `from pandas.tools.plotting` by `from pandas.plotting`
- What is a confusion matrix? For SVM RBF, change C and gamma parameters.

Pre-requisite for previous assignment

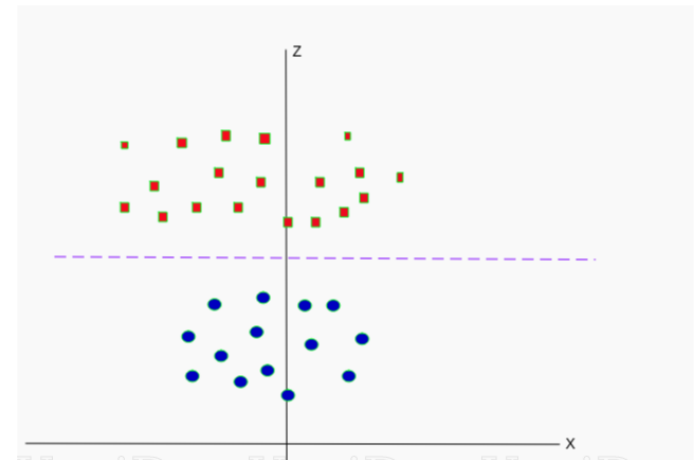
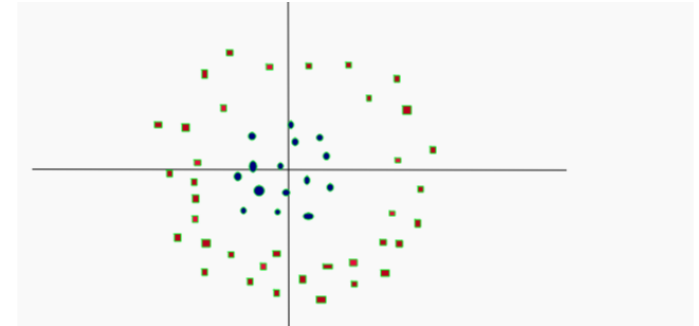
- ① <https://www.youtube.com/watch?v=P6kSc3qVph0>



- ① Part of PYTHON SKLEARN playlist from MachineLearnia
https://www.youtube.com/playlist?list=PLO_fdPEVIfKoHQ3Ua2NtDL4nmynQC8YiS

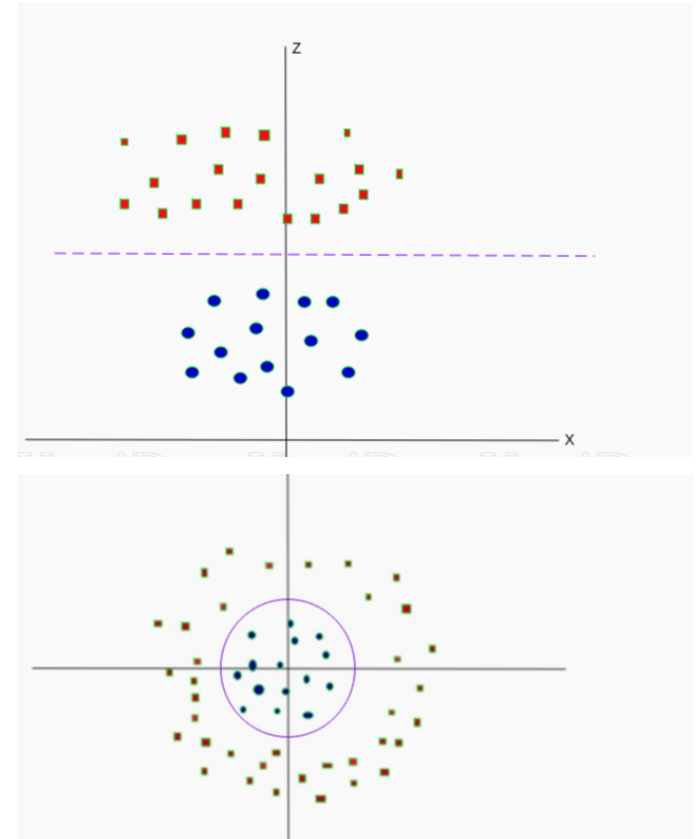
Pre-requisite for previous assignment

- ⦿ Non-linear SVM, consider the following nonlinearly separable dataset
- ⦿ This data can be converted to linearly separable data in higher dimension
- ⦿ Lets add one more dimension and call it z-axis: $z = x^2 + y^2$
- ⦿ z coordinate is the square of distance of the point from origin. Now the data is clearly linearly separable
- ⦿ Let the purple line separating the data in higher dimension be $z=k$, where k is a constant
- ⦿ Since, $z=x^2+y^2$ we get $x^2 + y^2 = k$; which is an equation of a circle!



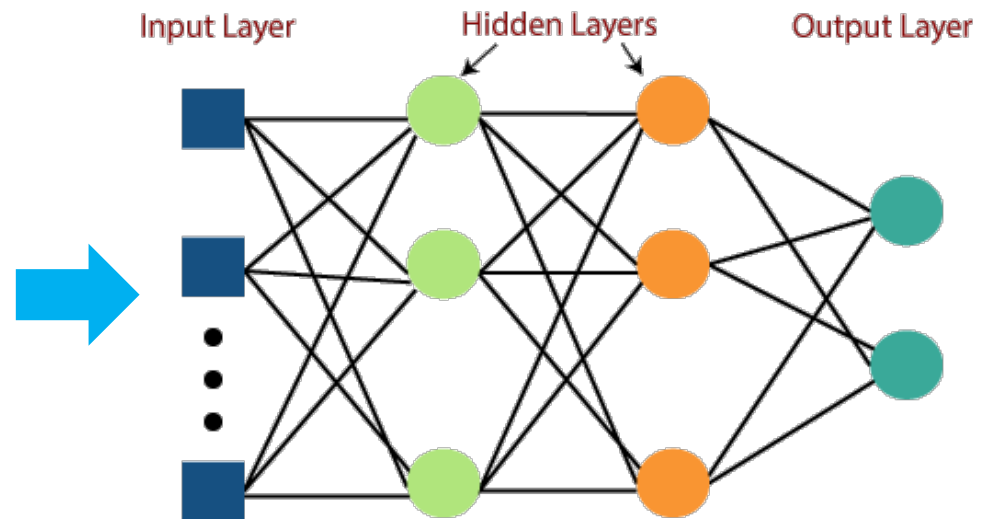
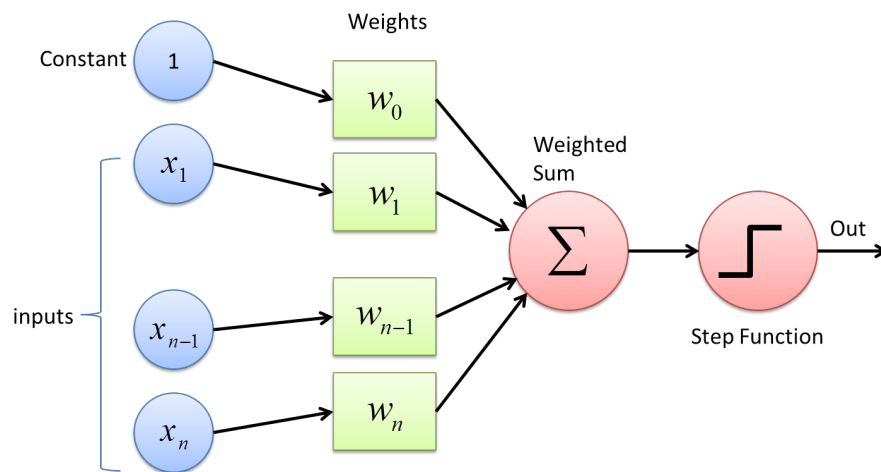
Pre-requisite for previous assignment

- ⦿ Since, $z=x^2+y^2$ we get $x^2 + y^2 = k$; which is an equation of a circle!
- ⦿ We can project this linear separator in higher dimension back in original dimensions using this transformation
- ⦿ But finding the correct transformation for any given dataset isn't that easy
- ⦿ Thankfully, we can use kernels in SKLEARN's SVM implementation to do this job
- ⦿ For instance, the Radial Basis Function (RBF) kernel
- ⦿ <https://scikit-learn.org/stable/modules/svm.html#parameters-of-the-rbf-kernel>



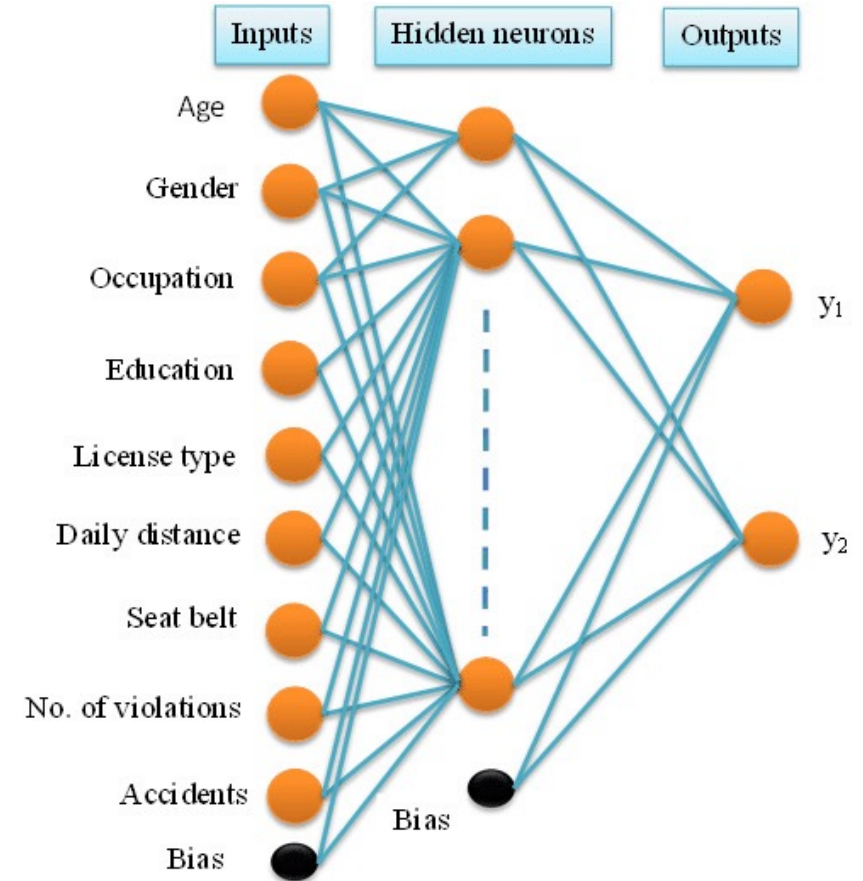
Multi-Layer Perceptron – MLP

- Multi-Layer Perceptrons (and so-called neural networks) can be considered as more general function approximators and they are able to distinguish data that is not linearly separable
- MLP have several neuron layers with so-called hidden layers
- Hidden layers are "hidden" to the final user who only "see" inputs and outputs

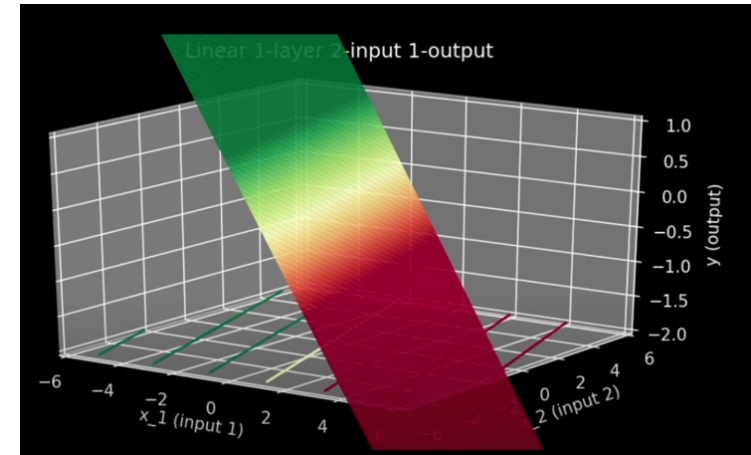
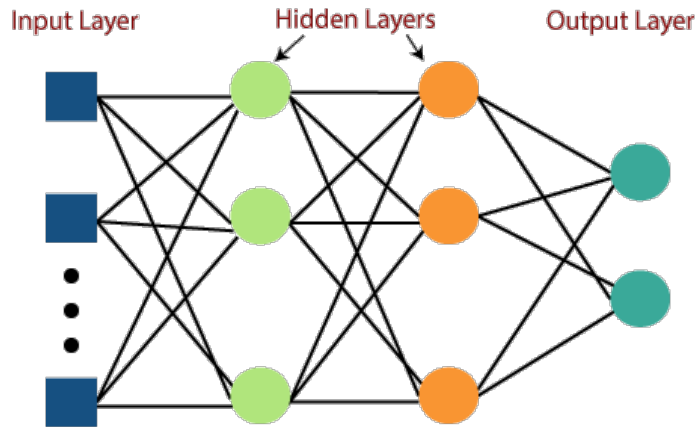


Towards decision-making!

- MLP are more general function approximators, taking a large number of features in consideration
- The complex classification can become a simple decision-making process!
- Here, should an insurance company sign with a driver or not!
- With more outputs, we could also have more final decisions such as the insurance price!



Again, what output function?



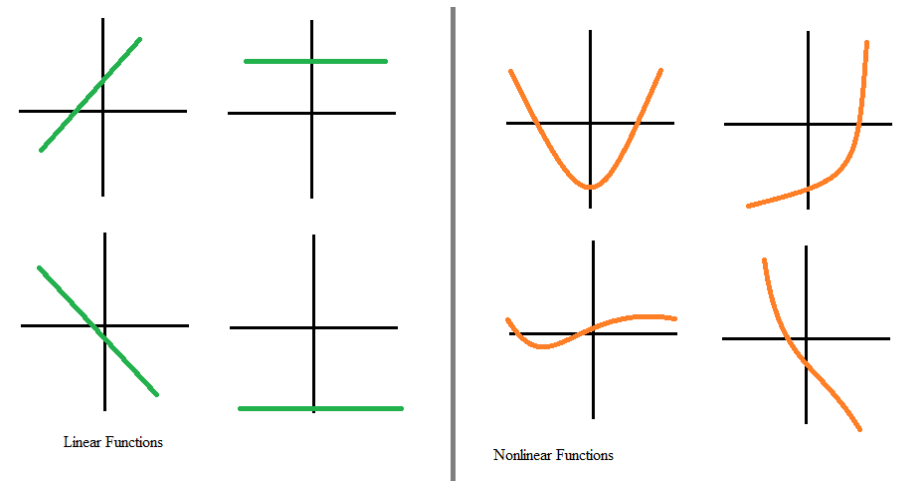
<https://www.youtube.com/watch?v=dPWYUELwldM>

- From SLP to MLP, regardless of the number of layers or weights, if outputs are linear, there is no way to have non-linear behavior!

$$\hat{y} = \Theta(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$

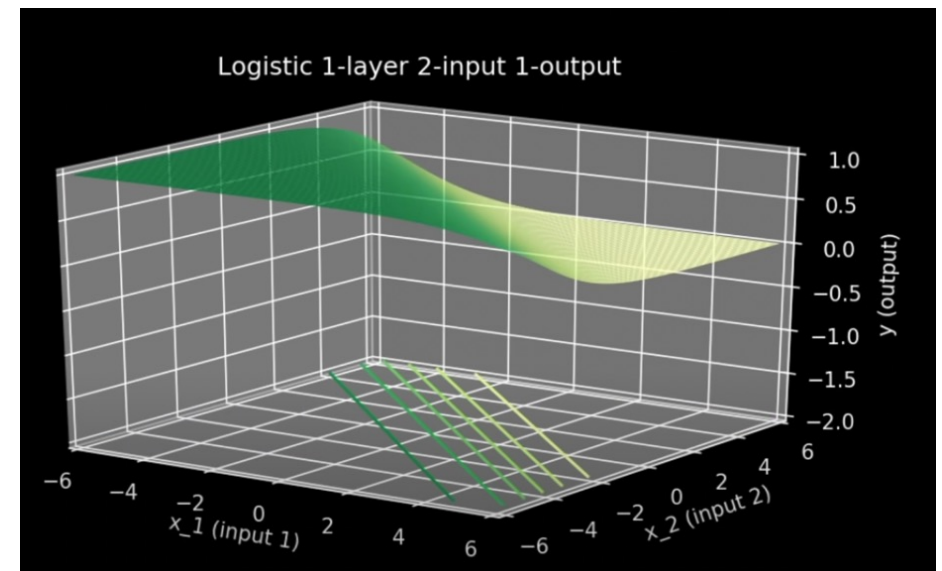
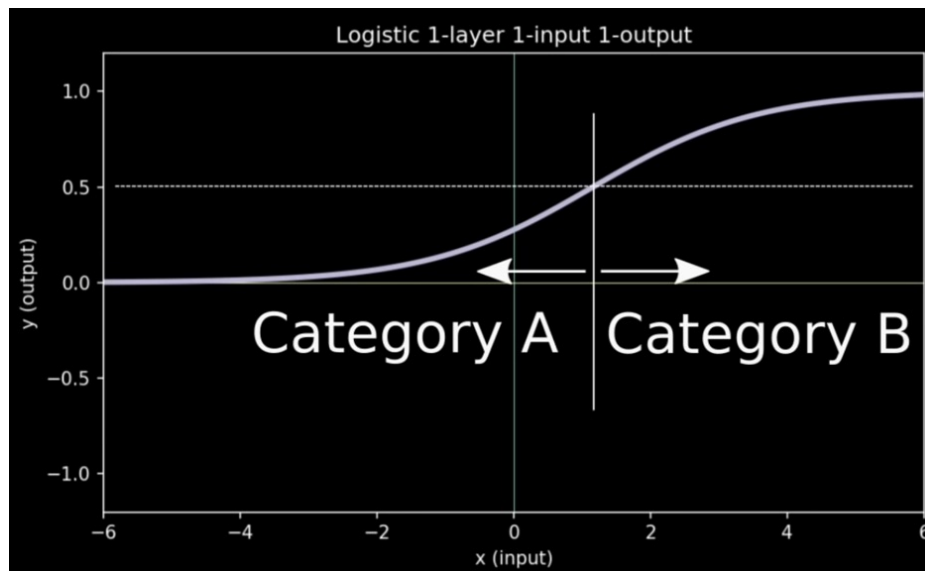
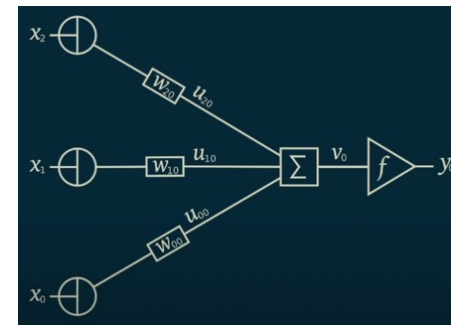
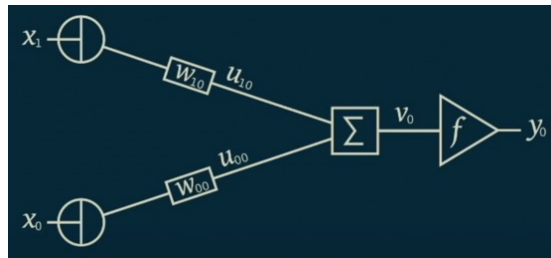
$$= \Theta(\mathbf{w} \cdot \mathbf{x} + b)$$

- This is why MLP usually have non-linear output function



Back to Logistic function

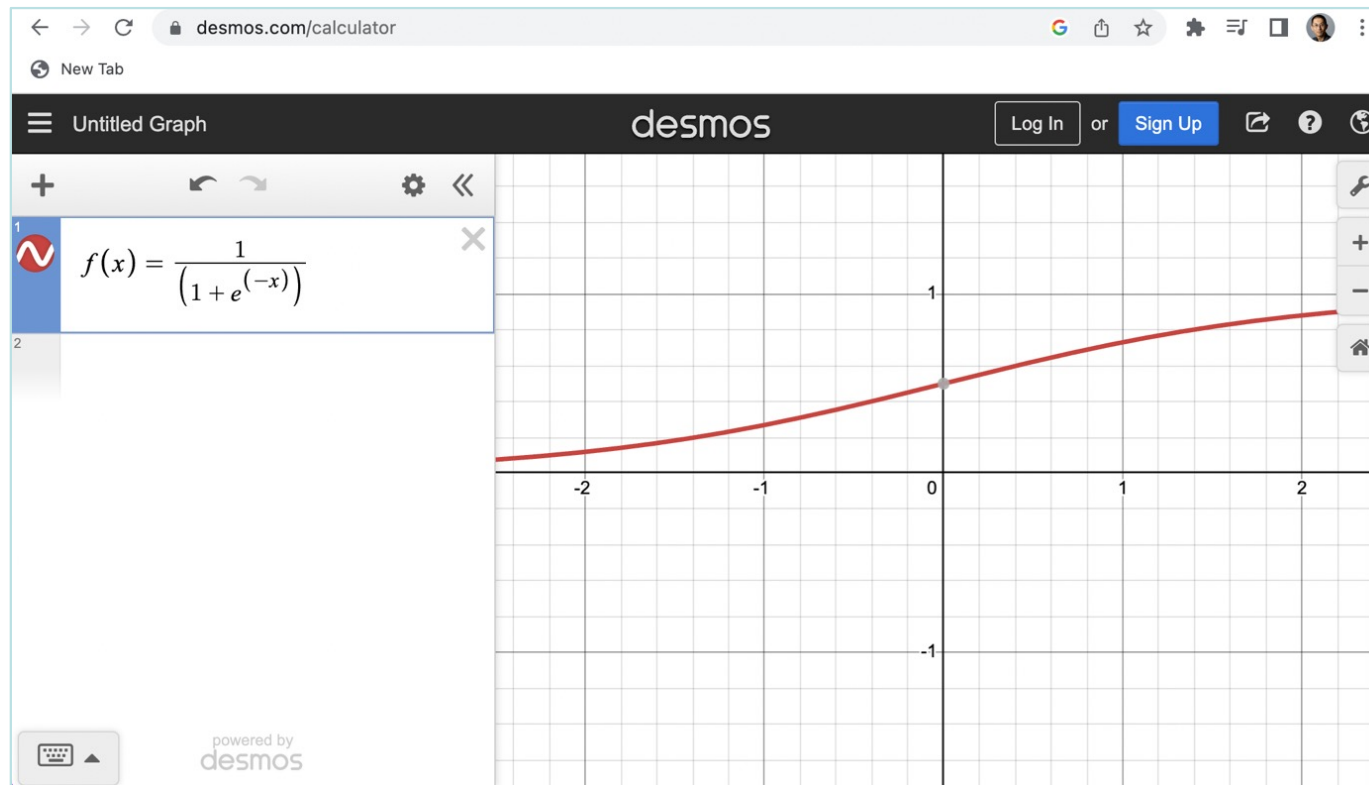
- ⦿ The Logistic function (Sigmoid) is non-linear and can separate 2 categories



<https://www.youtube.com/watch?v=dPWYUELwldM>

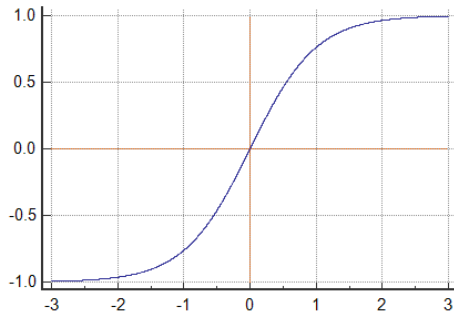
Online function plotting

① www.desmos.com/calculator

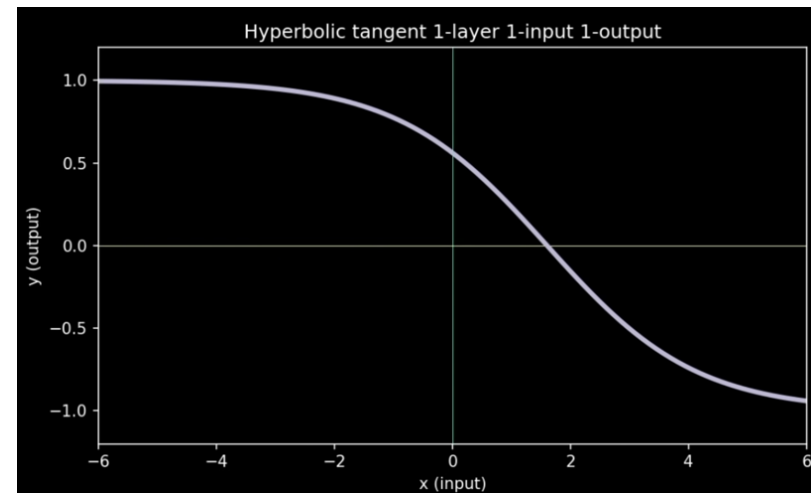
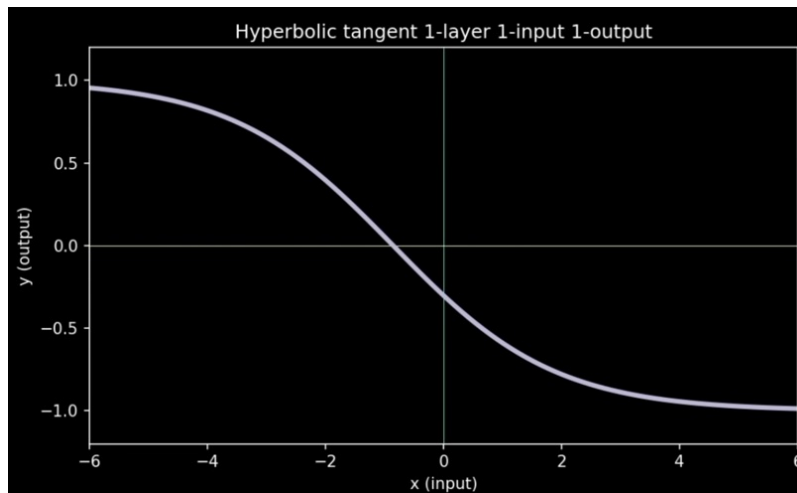
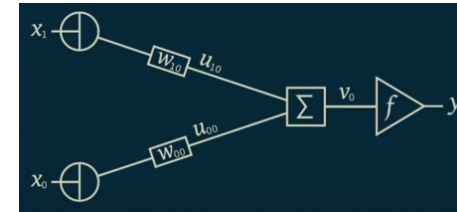


The hyperbolic tangent – tanh

- It is basically a shifted sigmoid neuron. It takes a real valued number and squashes it between -1 and +1



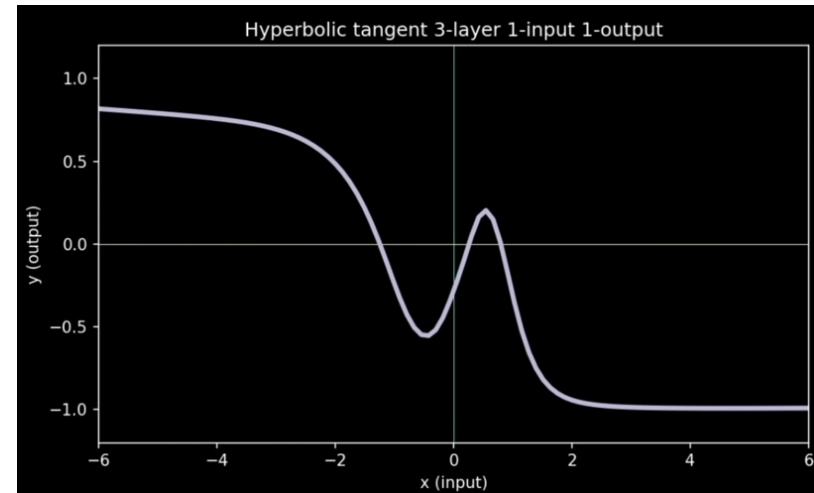
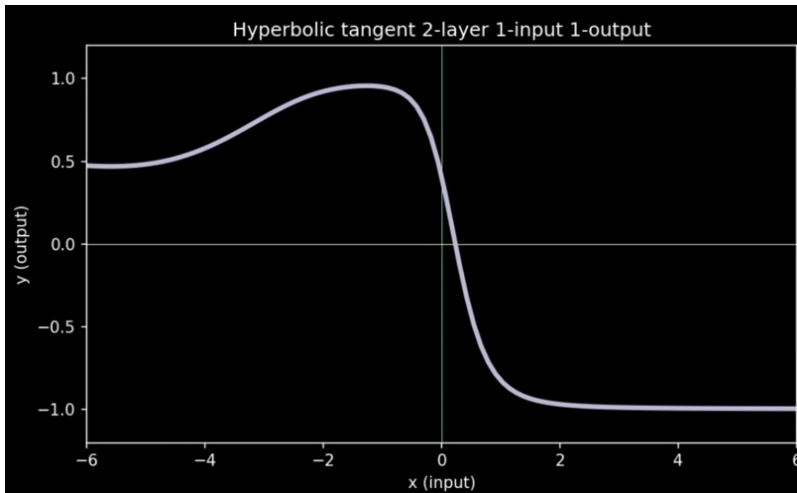
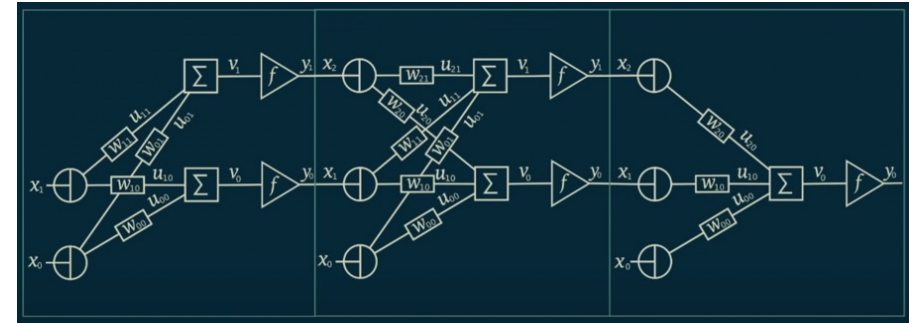
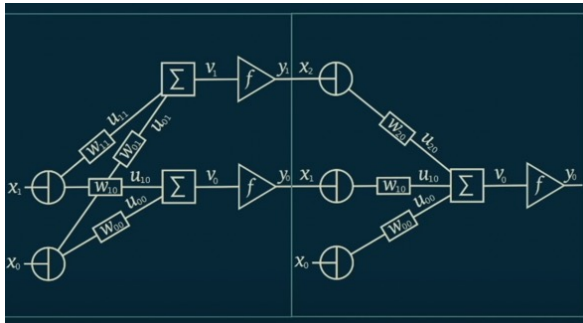
$$\tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} = \frac{2}{1 + e^{-2x}} - 1 = 2\text{sigmoid}(2x) - 1$$



<https://www.youtube.com/watch?v=dPWYUeLwldM>

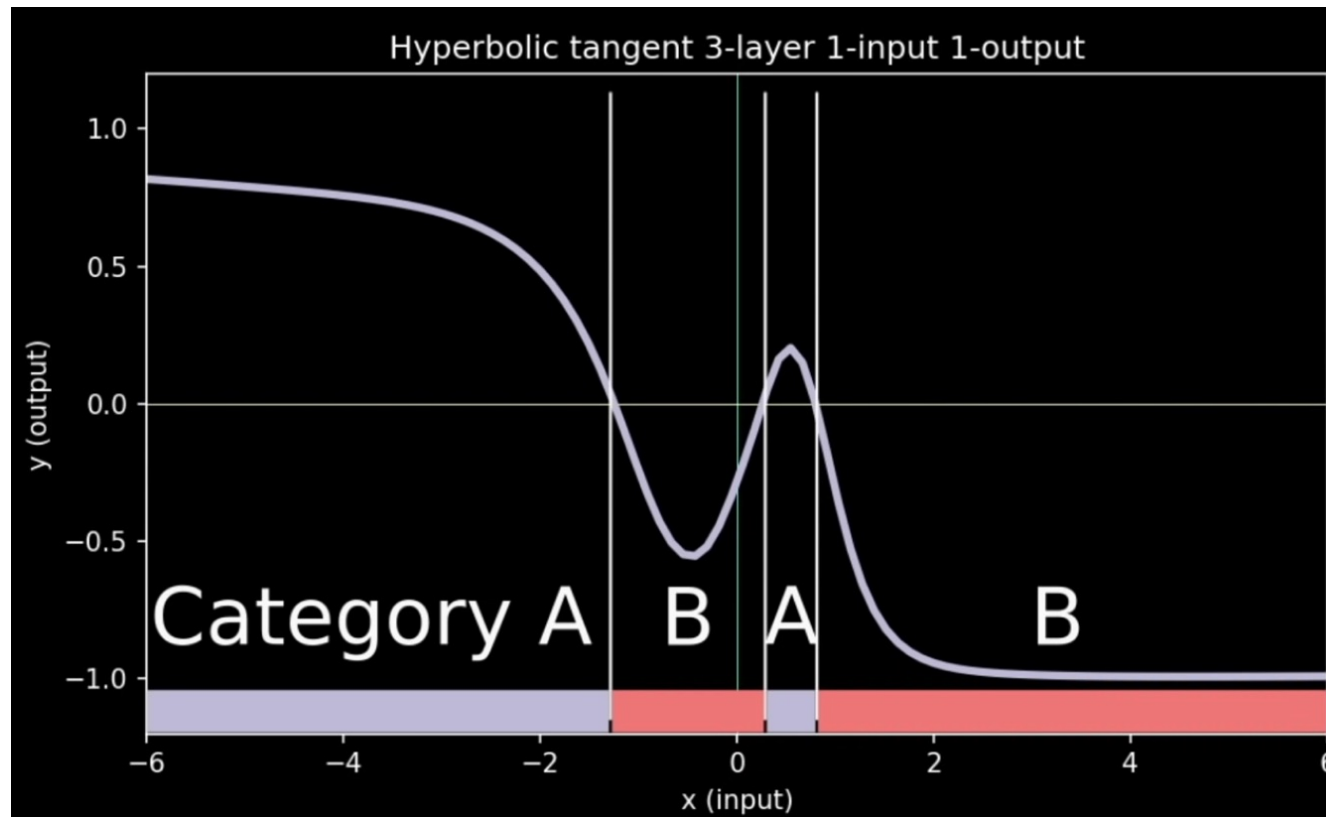
Tanh with several layers

- When adding additional layers, the output can be much more complex, allowing for non-linear classification



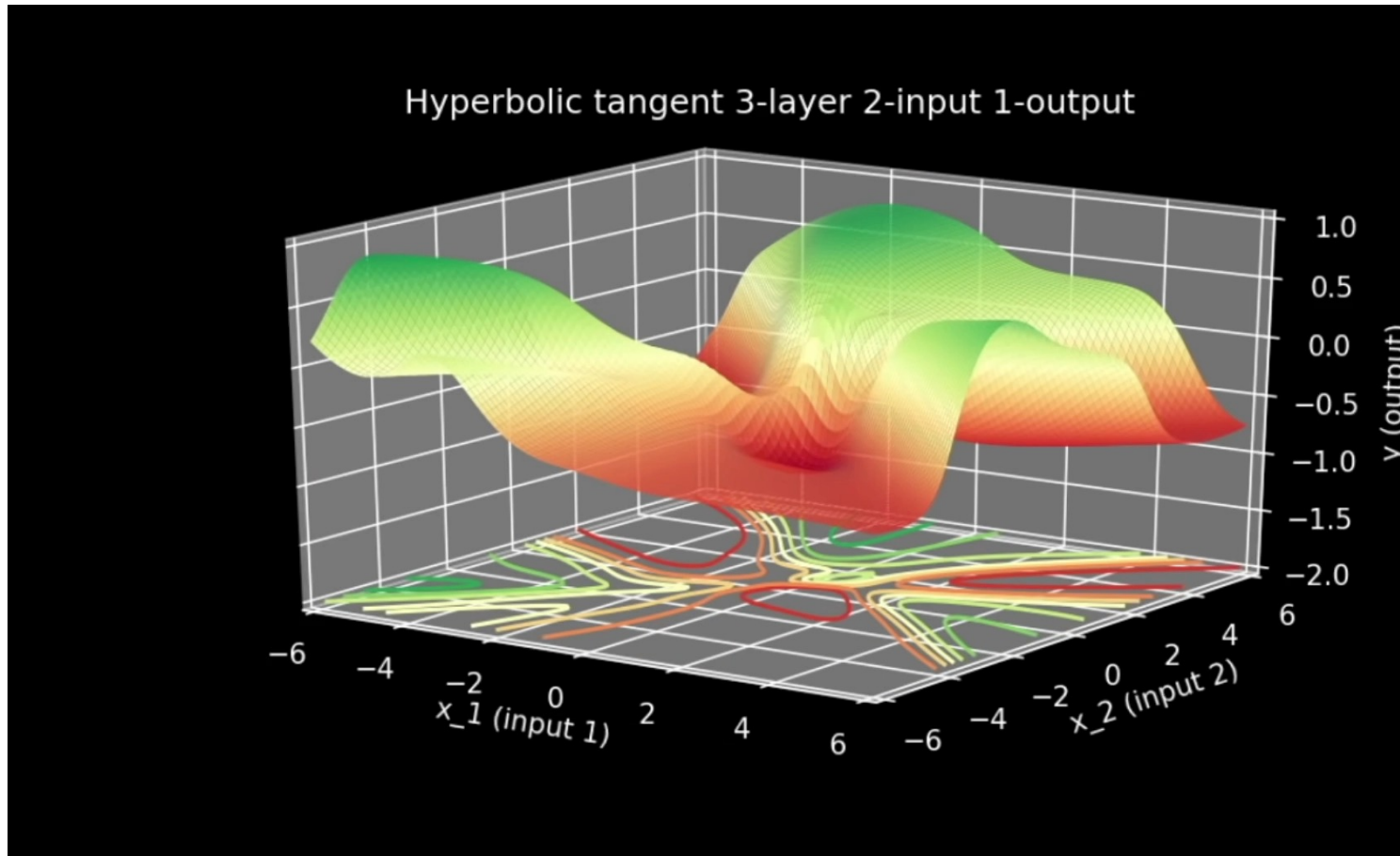
New classification possibilities?

- ⦿ The ability to create complex output enables the detection and the classification of a larger number of categories

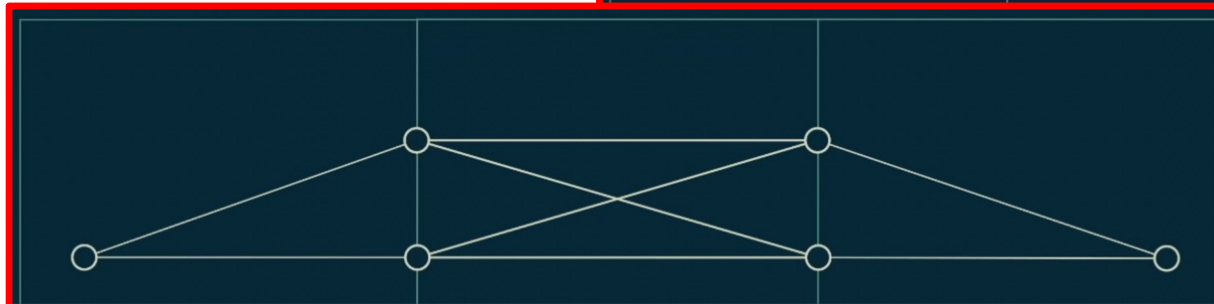
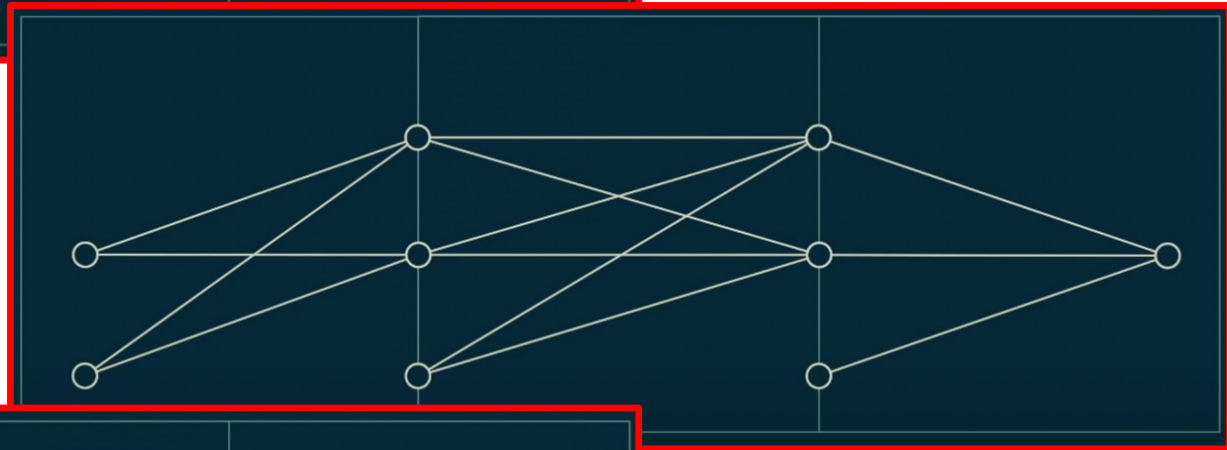
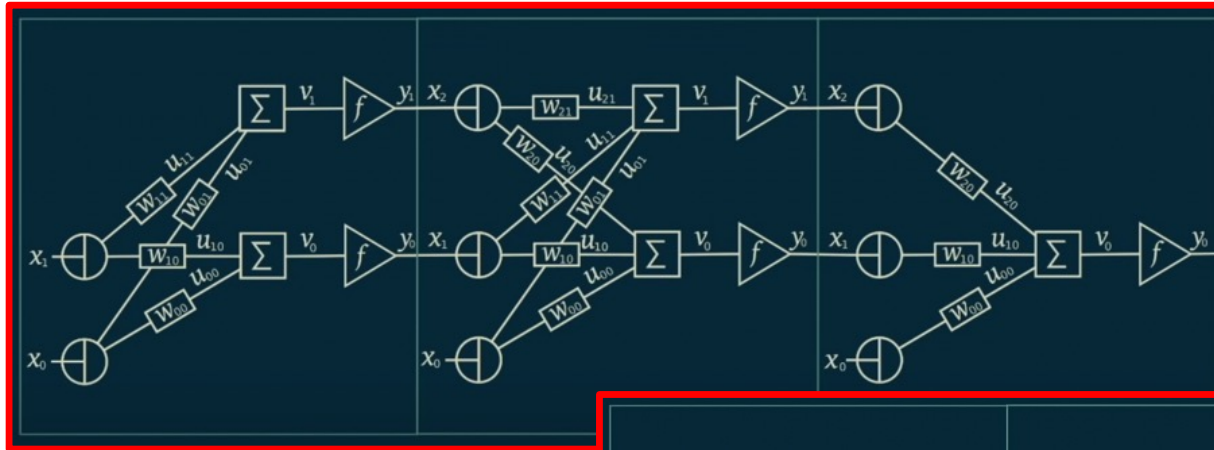


<https://www.youtube.com/watch?v=dPWYUELwldM>

See it in 3D – 3 layers, 2 inputs

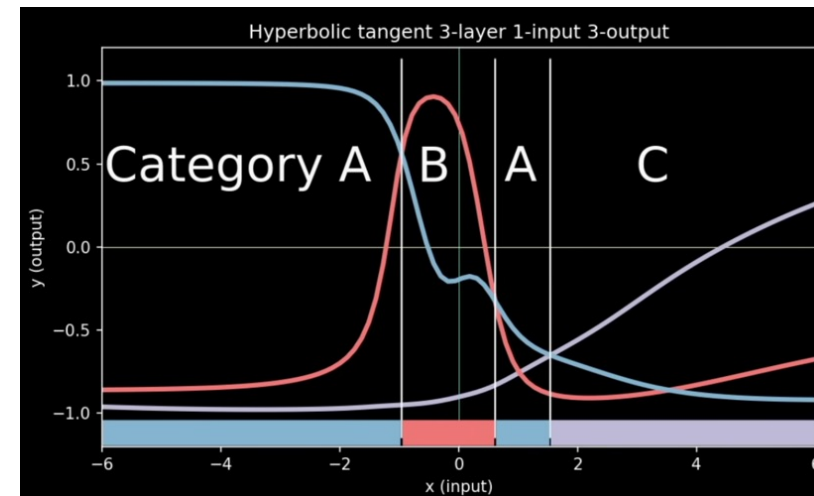
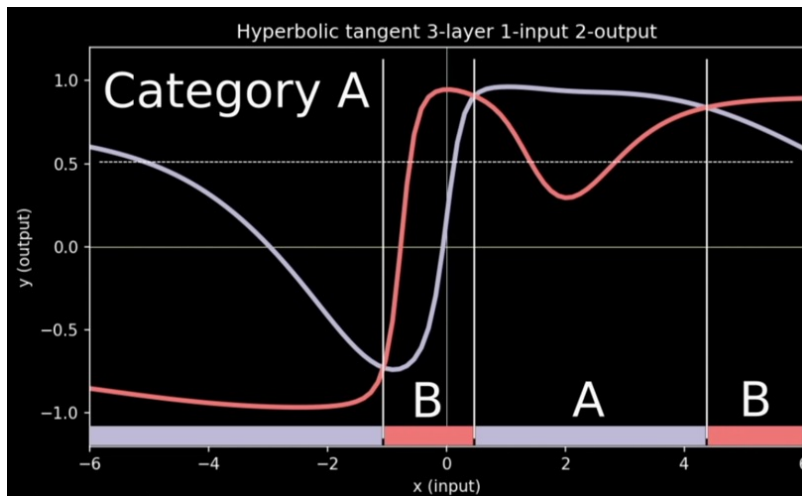
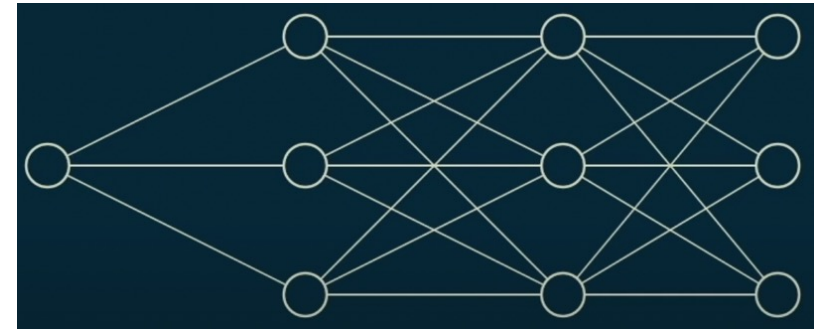
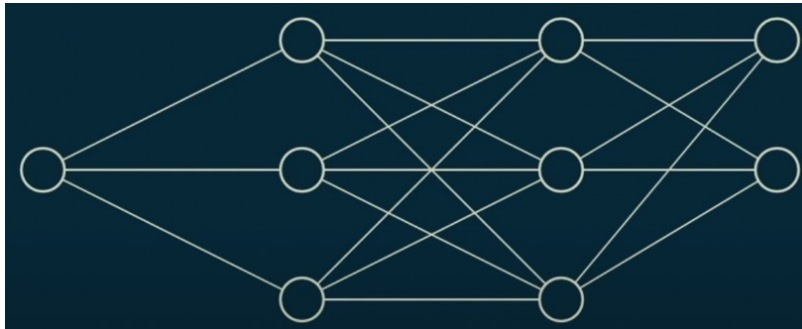


Simplifying the representation of MLP



Adding more outputs

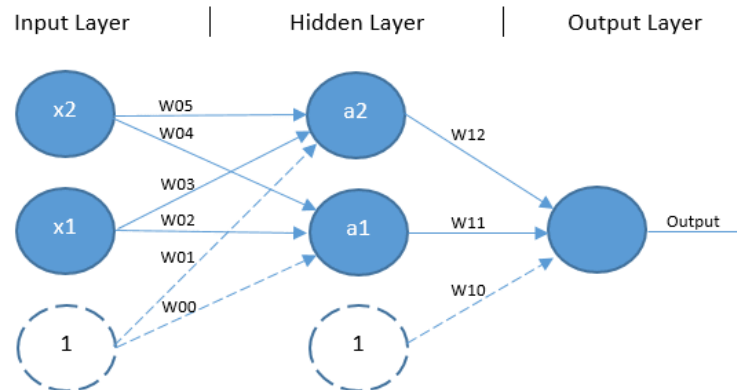
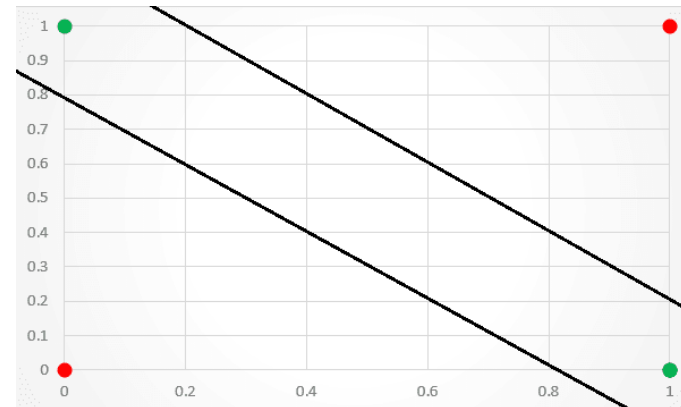
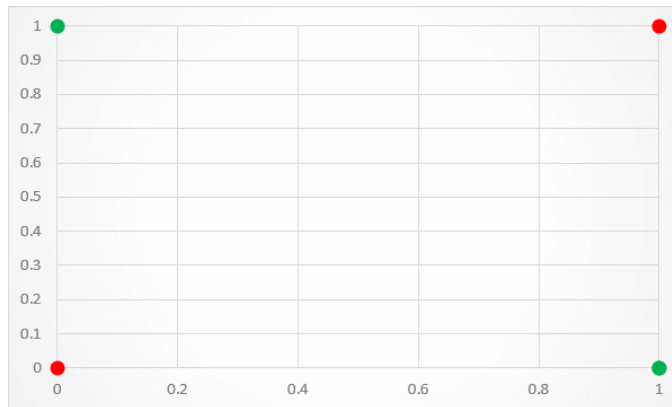
- 2 and 3 outputs with tanh activation fonction



<https://www.youtube.com/watch?v=dPWYUELwldM>

Back to the XOR problem

- ⦿ XOR is a nonlinearly separable problem so SLP won't work
- ⦿ MLP can solve nonlinearly separable problems so MLP should work

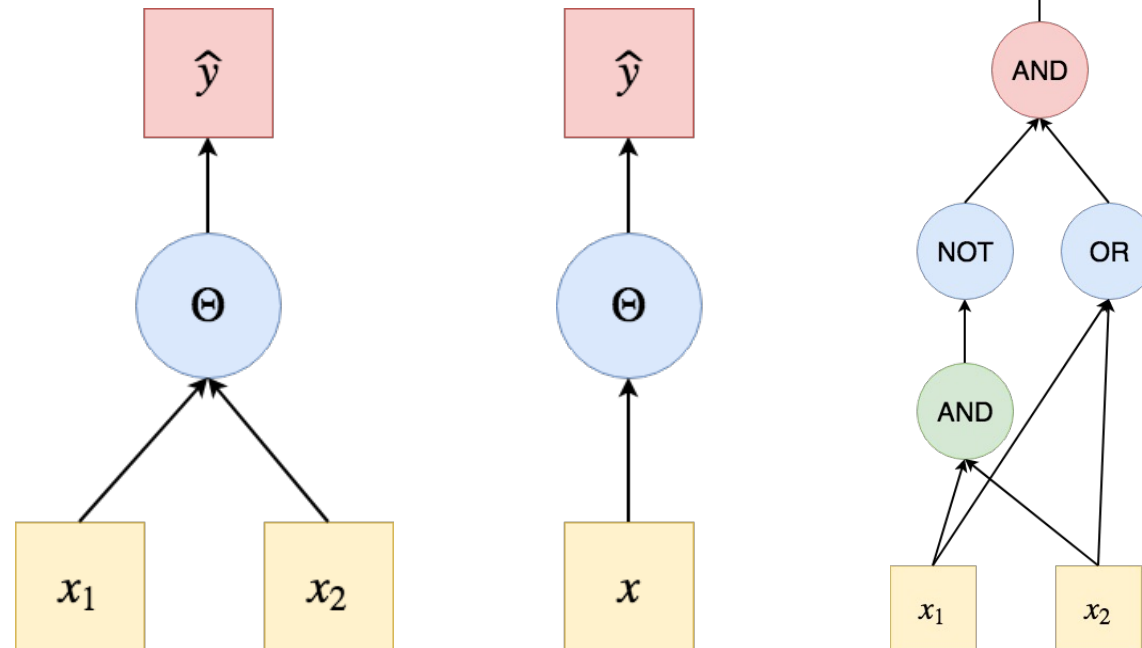


XOR: example 1

Assignment to do in lab

- ◉ <https://towardsdatascience.com/perceptrons-logical-functions-and-the-xor-problem-37ca5025790a>
- ◉ View it as combination of NOT, AND and OR gates

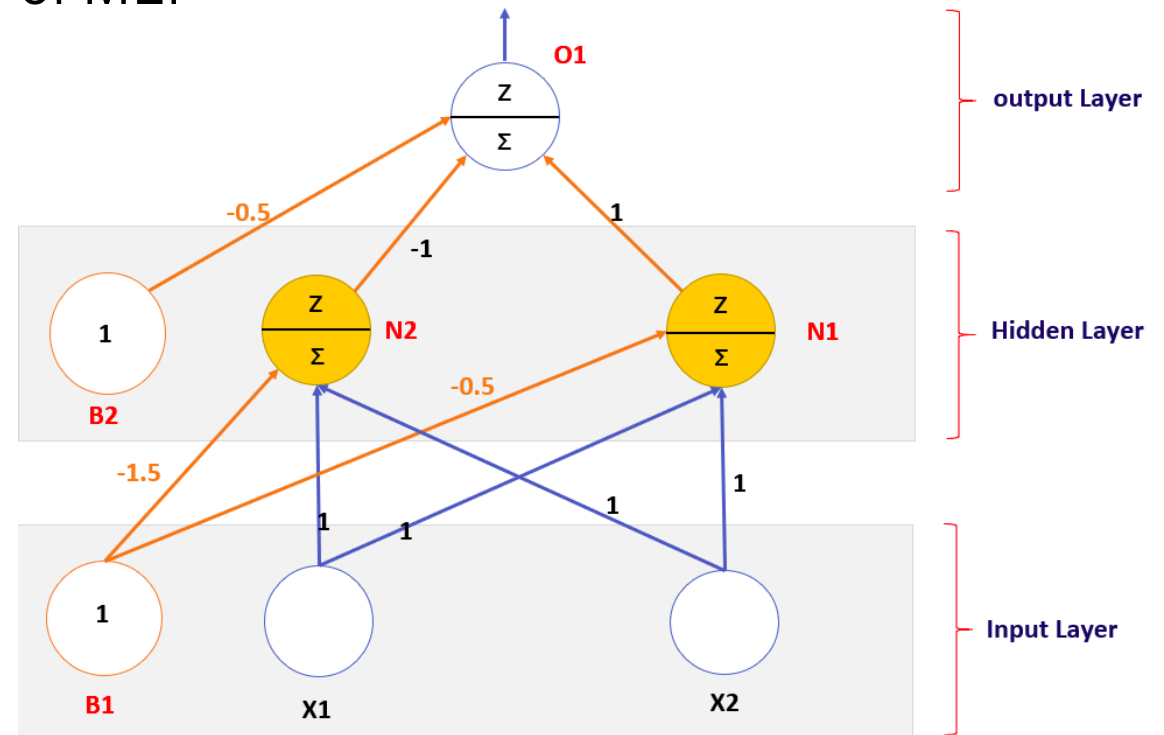
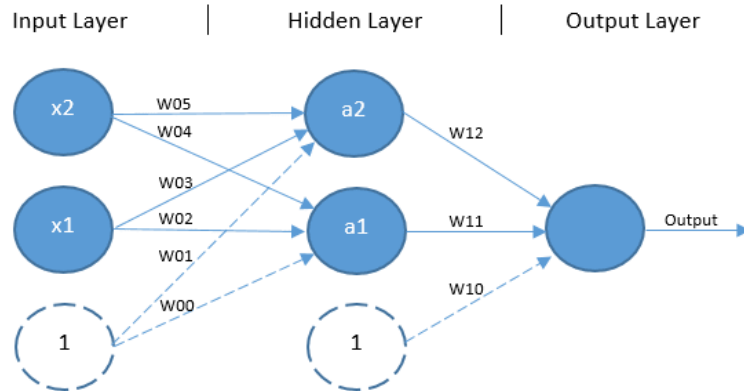
$$XOR(x_1, x_2) = AND(NOT(AND(x_1, x_2)), OR(x_1, x_2))$$



XOR: example 2

Assignment to do in lab

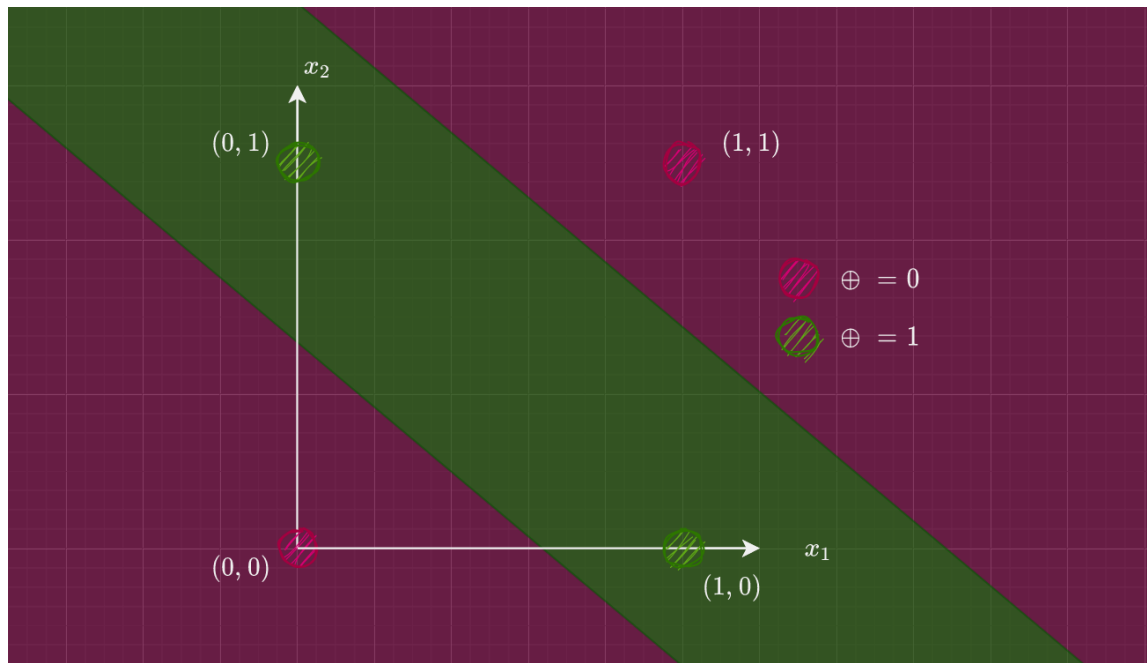
- <https://medium.com/analytics-vidhya/xor-gate-with-multilayer-perceptron-66e78671acd4>
- One step towards usage of MLP



XOR: example 3

Assignment to do in lab

- ① <https://towardsdatascience.com/how-neural-networks-solve-the-xor-problem-59763136bdd7>
- ① Finally, the MLP with training approach

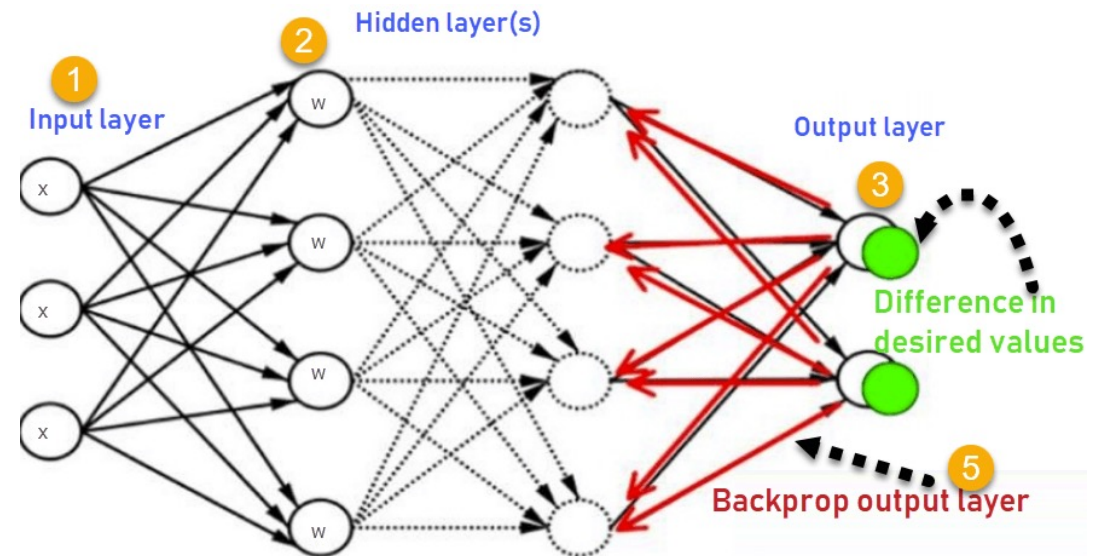


Learning in MLP?

- With more layers, there are more parameters to optimize, so training an MLP is obviously more complex and will take much much more time, if not done in an intelligent way!
- A fundamental method in neural network is back-propagation!

The errors are first calculated at the output units where the formula is quite simple (based on the difference between the target and predicted values), and then propagated back through the network in a clever fashion, allowing us to efficiently update our weights during training and (hopefully) reach a minimum.

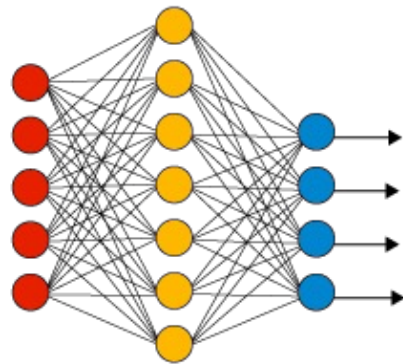
From <https://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks>



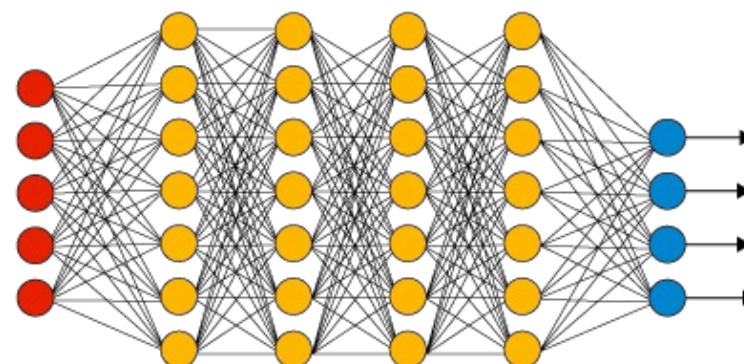
Deep Neural Networks

- ⦿ In the 1980s, most Artificial Neural Networks were single-layered due to the cost of computation and availability of data.
- ⦿ Nowadays is possible to afford more hidden layers, hence the moniker “Deep Neural Networks”.
- ⦿ Regained popularity since ~2006 and rebranded as Deep Learning (DL)

Simple Neural Network



Deep Learning Neural Network



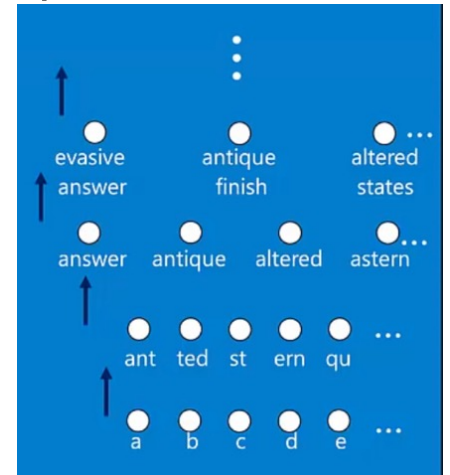
● Input Layer

● Hidden Layer

● Output Layer

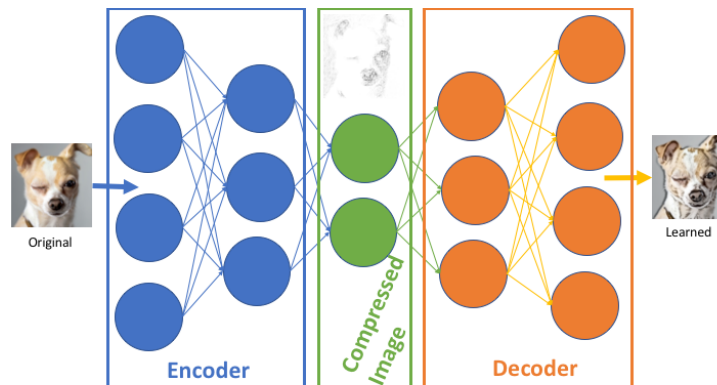
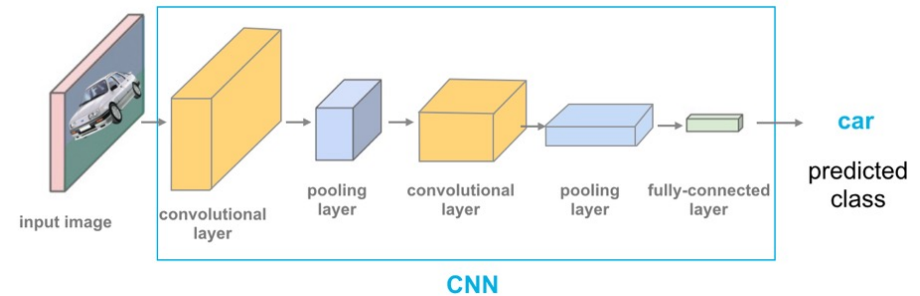
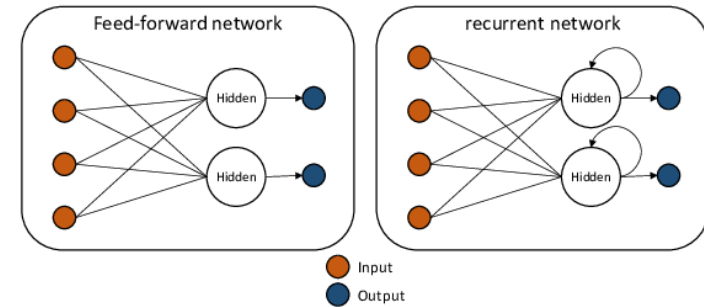
Hidden layers

- By the universal approximation theorem, a single hidden layer network with a finite number of neurons can be trained to approximate an arbitrarily random function. In other words, a single hidden layer is powerful enough to learn any function.
- That said, we often learn better in practice with multiple hidden layers (i.e., deeper nets)
- Hidden layers in a NN allows to create/store internal abstract representation of the training data
- The higher layers are “building” new abstractions on top of previous layers
- From <https://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks>



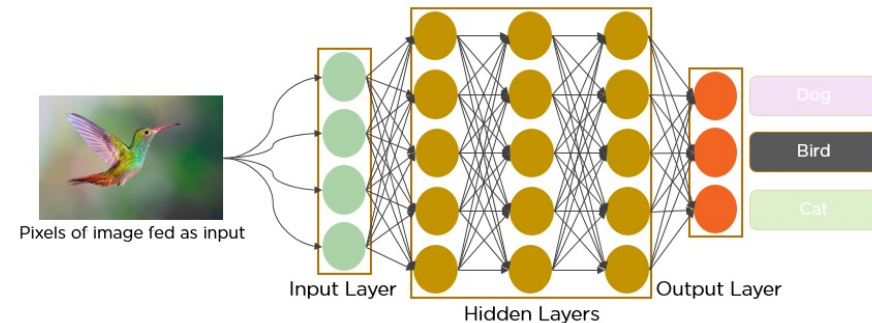
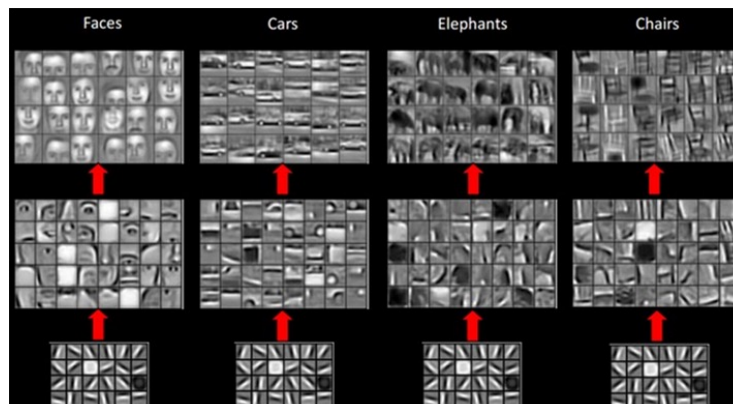
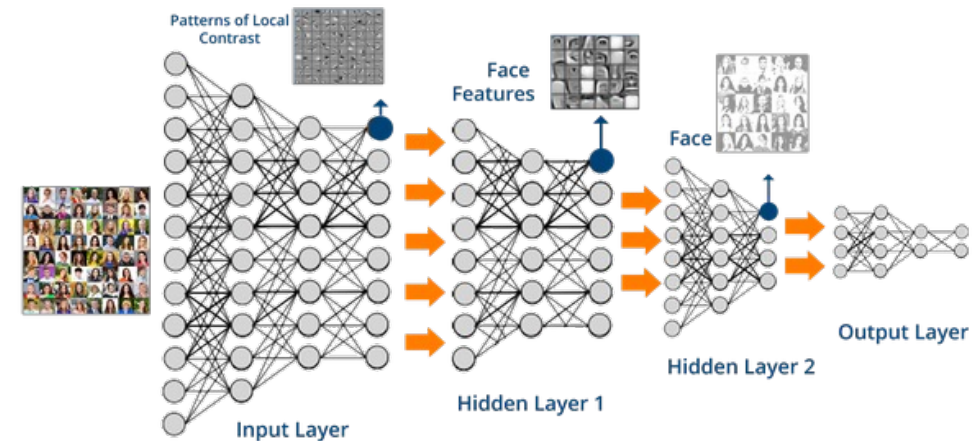
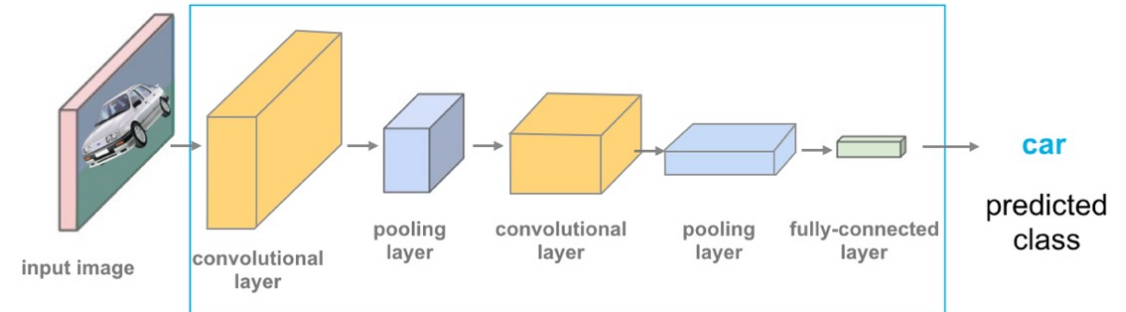
Types of Deep Neural Networks

- Feedforward Neural Networks (**FFNs, ANNs** or **NNs**)
- Recurrent Neural Networks (**RNNs**)
- Convolutional Neural Networks (**CNNs**)
- Autoencoder Neural Networks (**AEs**)



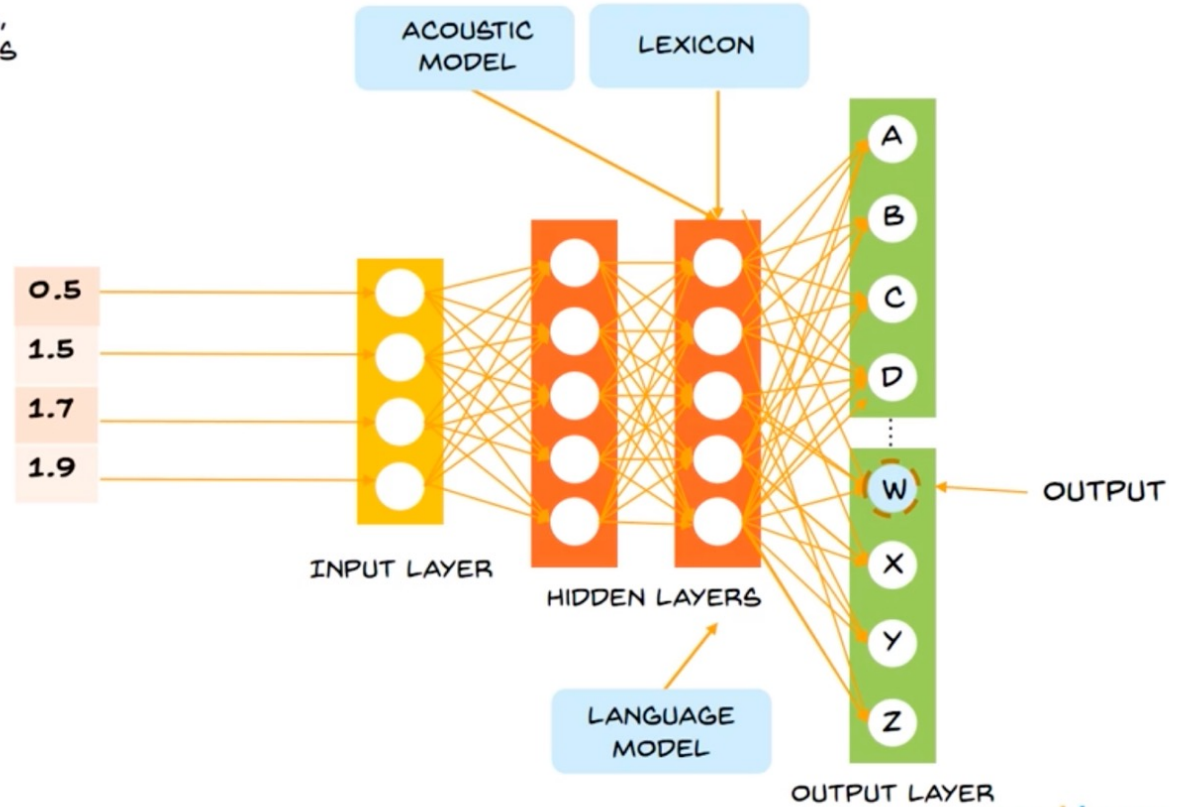
Convolutional Neural Networks

- Contain five types of layers
- Each layer has a specific purpose, like summarizing, connecting or activating
- CNN are good at image classification and object detection

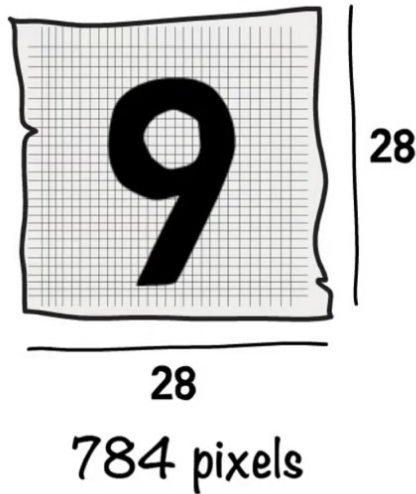


Examples for Deep Learning (1)

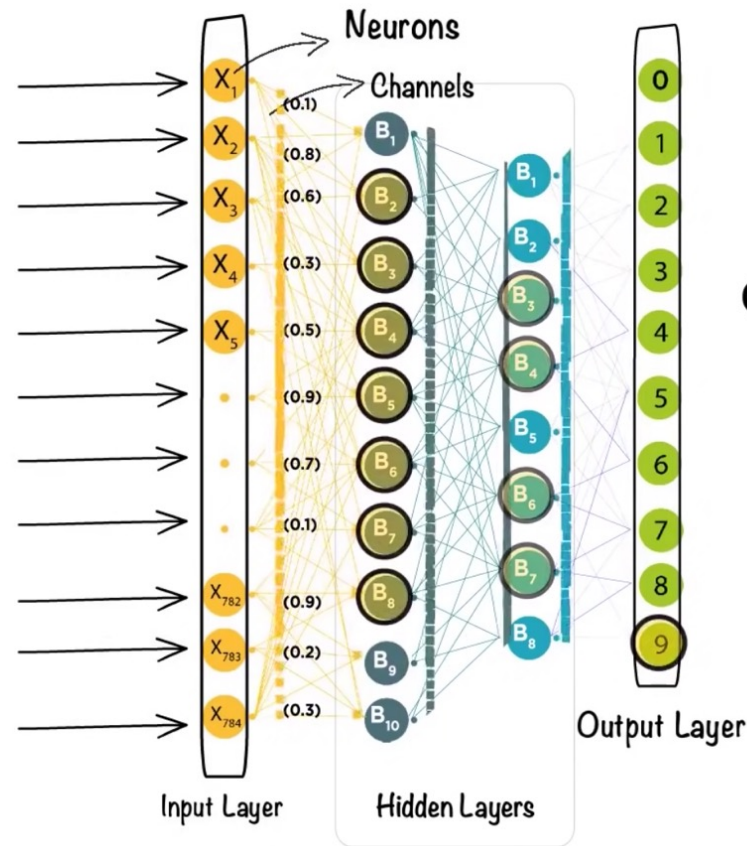
WITH THE HELP OF THESE MODELS, THE NETWORK PREDICTS THE WORDS THAT YOU ARE TRYING TO SAY!



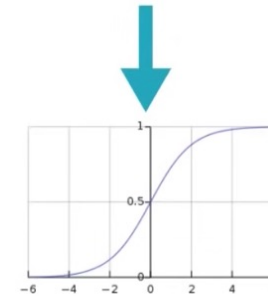
Examples for Deep Learning (2)



Deep Learning



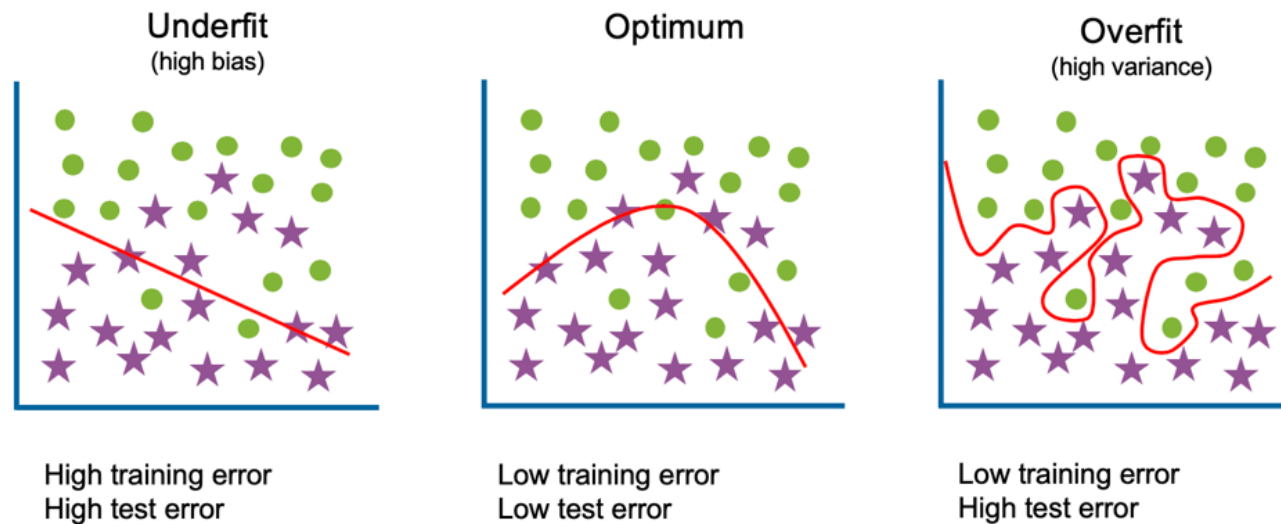
$$B_1 + (X_1 * 0.1 + X_2 * 0.8)$$



Activation Function

Overfitting?

- Overfitting describes the phenomenon of fitting the training data too closely, maybe with hypotheses that are too complex. In such a case, your learner ends up fitting the training data really well, but will perform much, much more poorly on real examples.



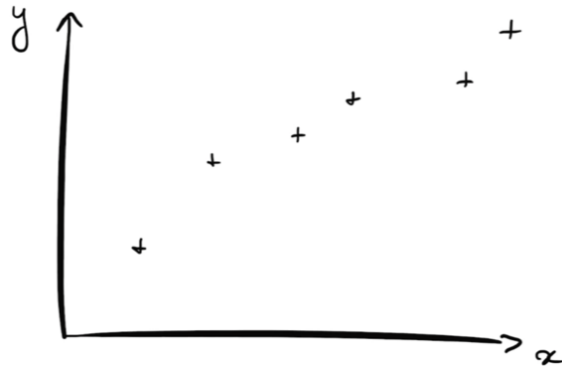


Gradient Descent

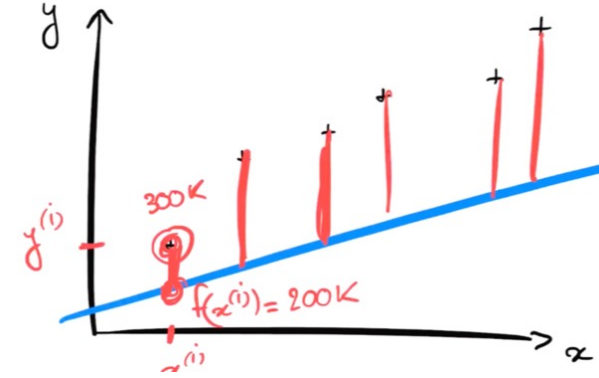
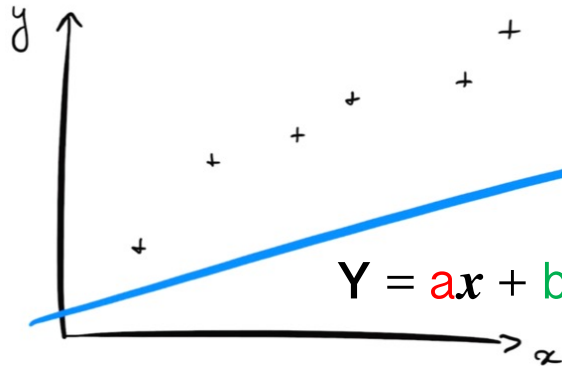
in

AI

Back to the cost function in regression



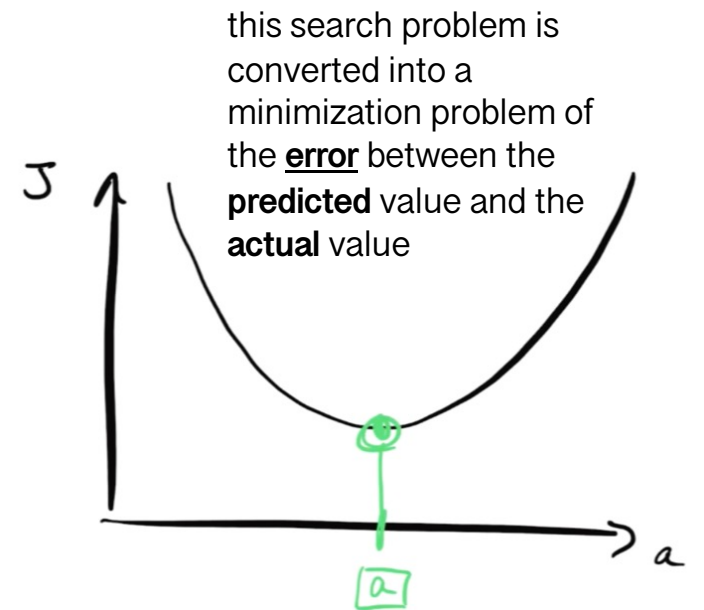
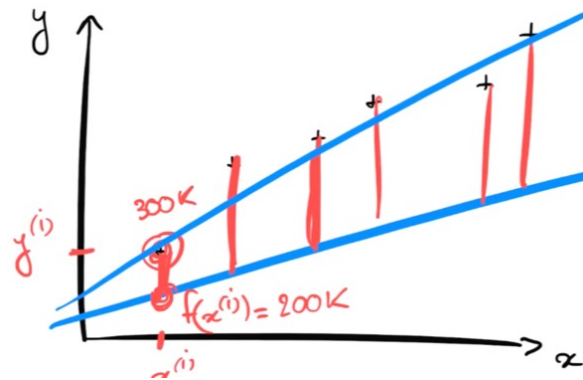
1. Dataset: (x, y) $m = 6, n = 1$ $x^{(i)}$



$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$

A **cost function** will help to figure out the best possible values for **a** and **b** which would provide the best *regression line* for data points

Here, Mean Squared Error (MSE)

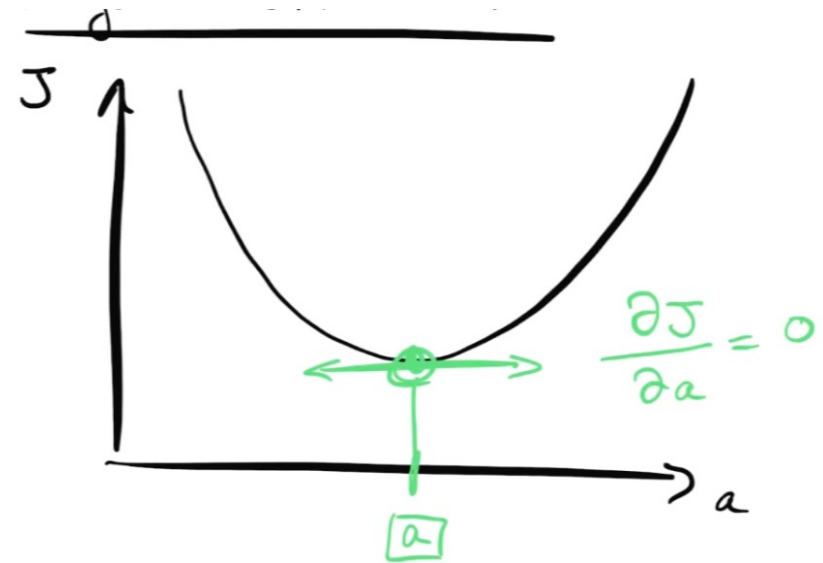


How to find the minimum?

- Exact methods can be used to find the minimum of the cost function

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$

- But these exact methods needs a lot of complex maths
 - What if the size of the dataset is large, e.g. millions of points?
 - What if we have multiple variable, e.g. multivariate regression?
- Then these complex math operations will take a long, long, long time even for state-of-the-art supercomputers!



Lost in the mountain?

- ⦿ Finding the lowest point...at the bottom of the valley



**Get to a safe
place, at the
bottom of the
valley**



Pictures from https://www.youtube.com/watch?v=rcl_YRyoLIY (Machine Learnia)

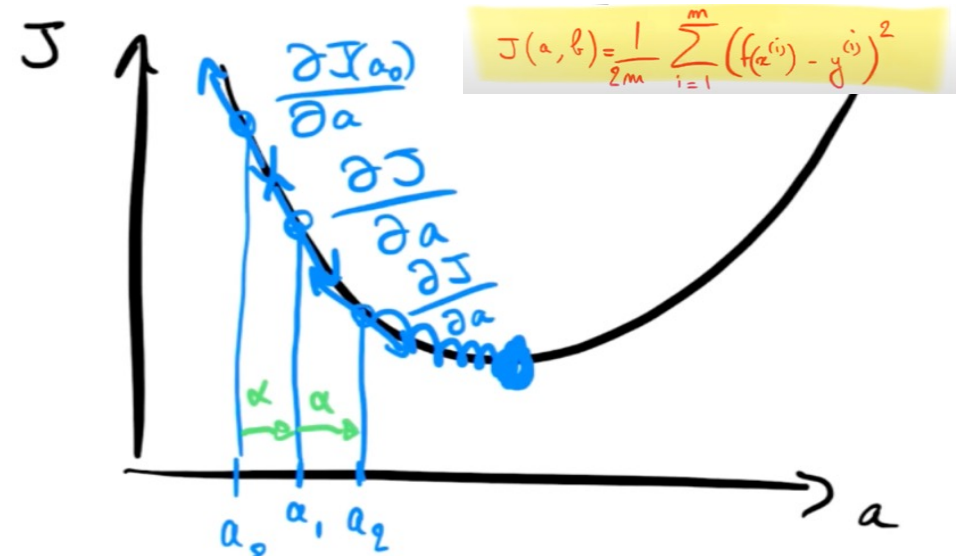
Principle behind Gradient Descent

- From your position, look around you and find the direction with the highest slope
- Walk for 300m in that direction
- Re-iterate until you reach the bottom



Machine version

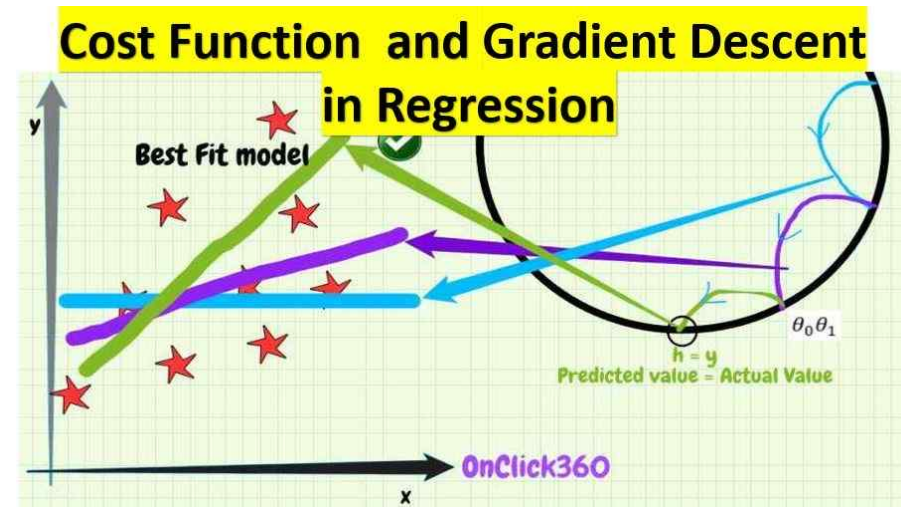
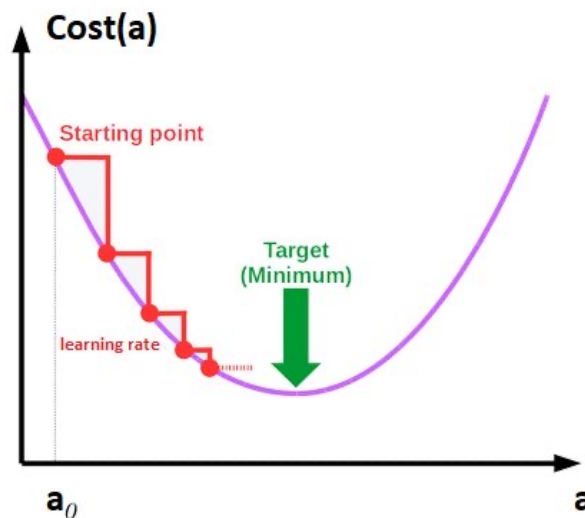
- Start at a_0 and compute the slope, i.e. the derivative
- Advance from a_0 to a_1 ($\alpha=300m$)
- Re-iterate until derivative is zero



Gradient Descent for linear regression

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$

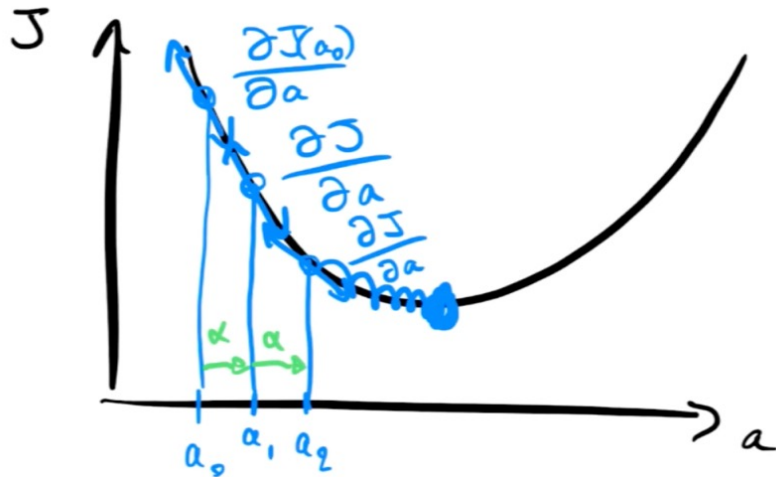
- Gradient Descent is a method of updating **a** and **b** to reduce the error (Cost Function)
- The idea is to start with arbitrary values for **a** and **b** then change these values iteratively to reduce the cost
- Gradient Descent helps on how to change the values



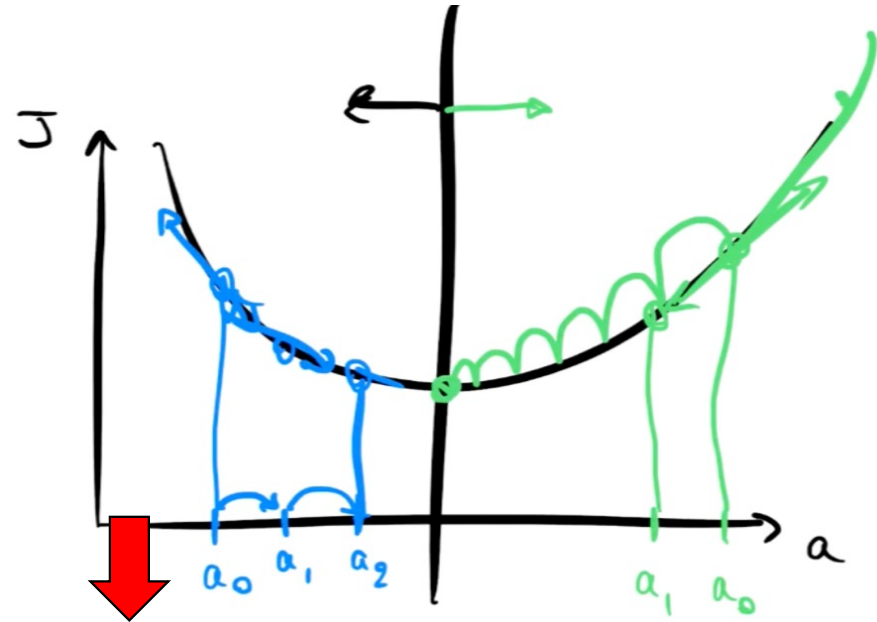
<http://aishelf.org/gradient-descent/>

Why is it more "intelligent"?

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$



$$a_{i+1} = a_i - \alpha \frac{\partial J(a_i)}{\partial a}$$



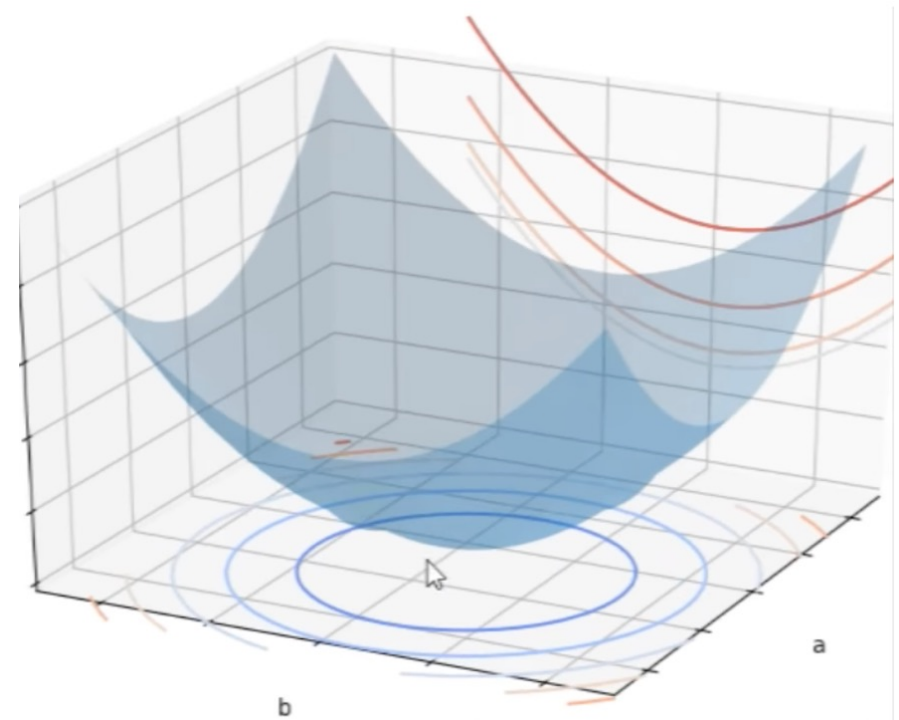
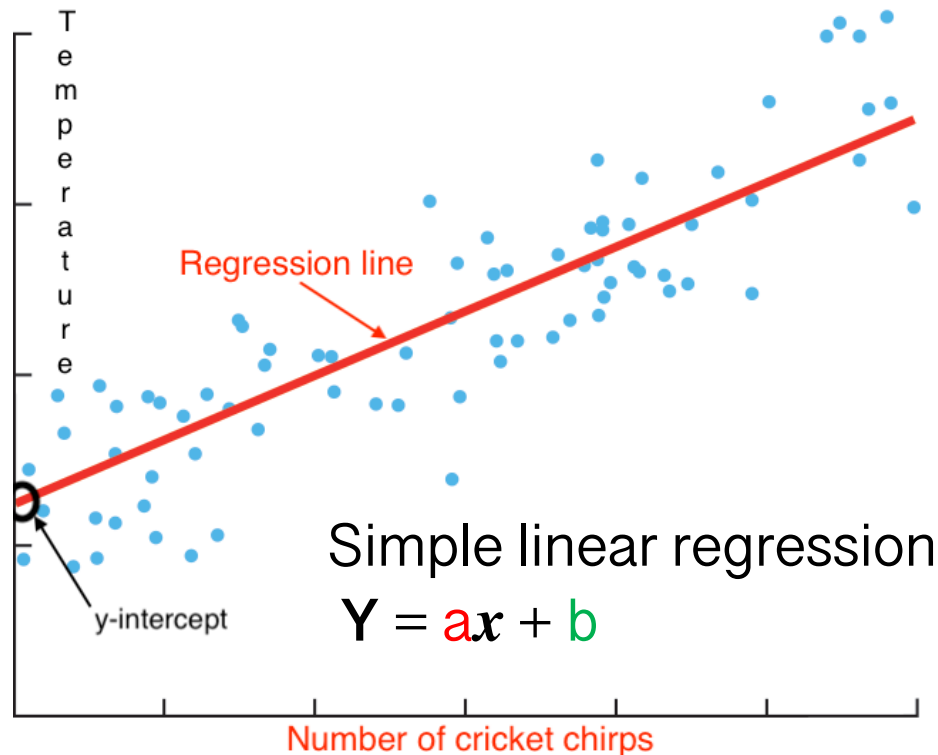
$$\frac{\partial J(a_i)}{\partial a} < 0, \text{ so } -\alpha \frac{\partial J(a_i)}{\partial a} > 0$$

$$\text{So } a_1 > a_0$$

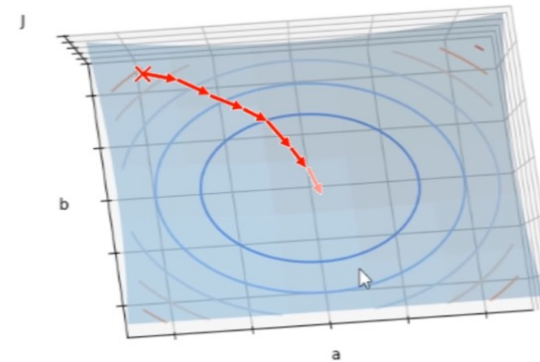
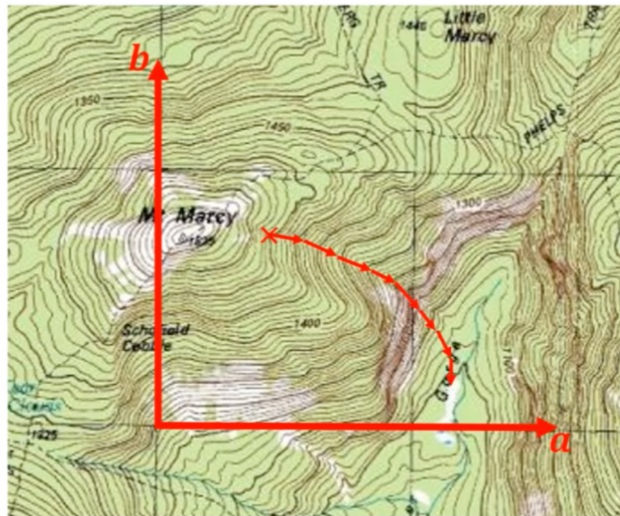
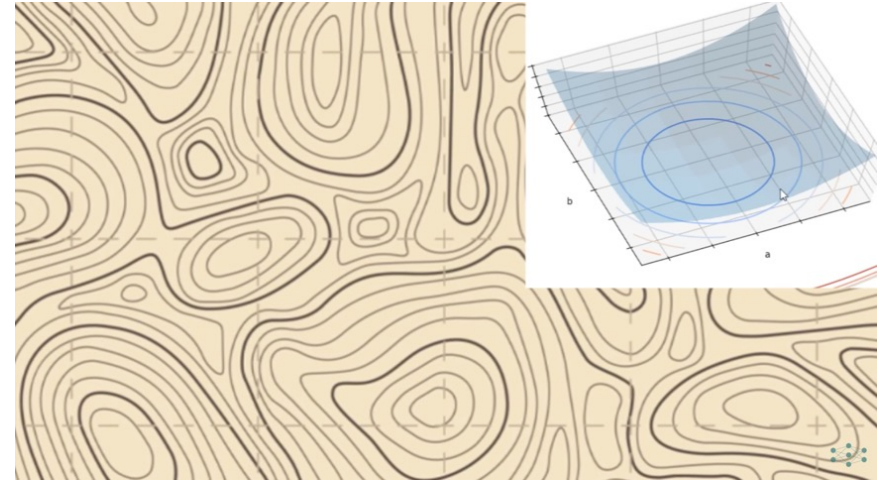
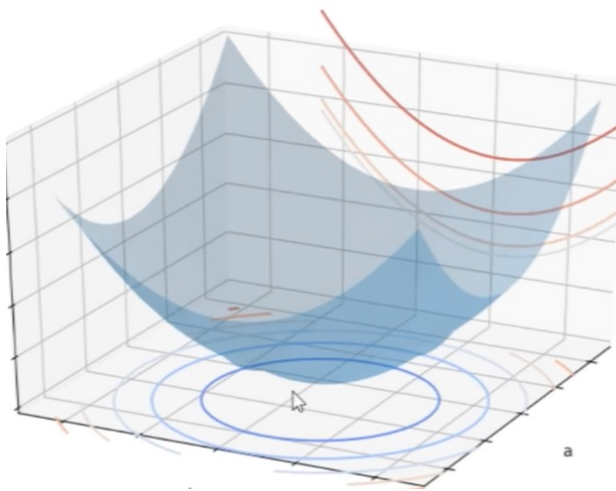
The algorithm converges automatically!

Why do we need that "intelligence"?

- With a simple line, we need to search for both a and b
- That already gives a large surface!
- So the search space can be huge!

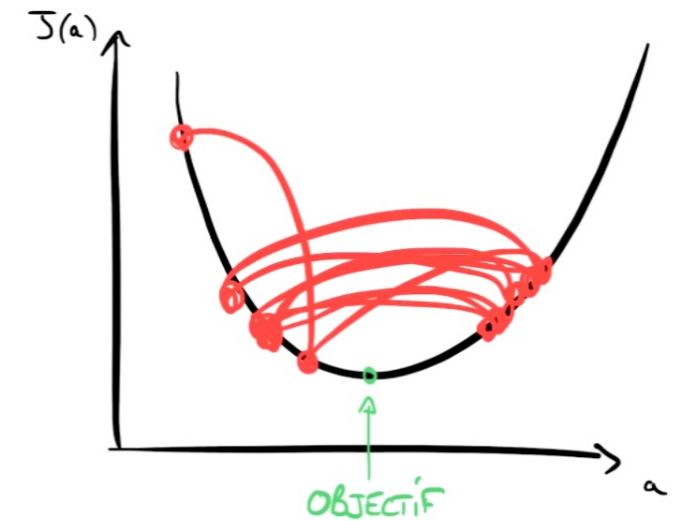
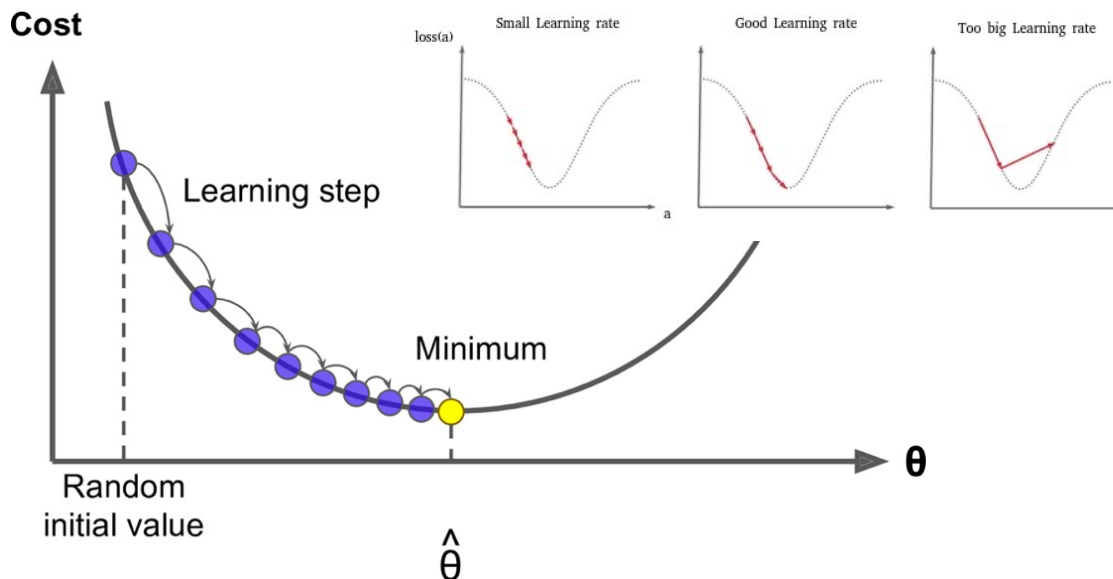
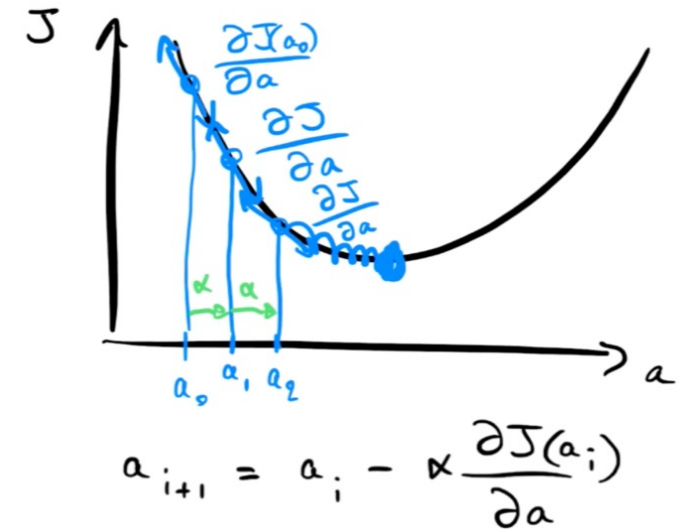


Gradient Descent ~ outdoor map !



Learning Rate α – hyperparameter

- ⦿ A smaller *learning rate* could get closer to the *minima* but takes more time to reach the *minima* (300m \rightarrow 50m)
- ⦿ A larger *learning rate* converges sooner but there is a chance that you could overshoot the *minima* (300m \rightarrow 800m)

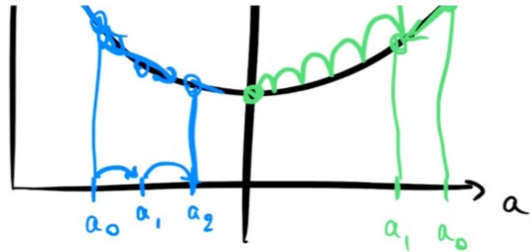


Equation of GD for linear regression

$$f(a,b) = ax + b - y$$

$$g(f) = f^2$$

$$(g \circ f)' = f' \cdot g'(f)$$



$$\left\{ \begin{array}{l} a = a - \alpha \frac{\partial J}{\partial a} \\ b = b - \alpha \frac{\partial J}{\partial b} \end{array} \right.$$

$$J(a,b) = \frac{1}{2m} \sum (ax + b - y)^2$$

$$\frac{\partial J}{\partial a} = \frac{1}{m} \sum \alpha(ax + b - y)$$

$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum (ax + b - y)$$

$$\left\{ \begin{array}{l} a = a - \alpha \frac{\partial J}{\partial a} \\ b = b - \alpha \frac{\partial J}{\partial b} \end{array} \right.$$

- Dataset : (x, y) avec m exemples
- Modèle : $f(x) = ax + b$
- Fonction Coût : $J(a, b) = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$
- Gradients :

$$\frac{\partial J(a, b)}{\partial a} = \frac{1}{m} \sum_{i=1}^m x^{(i)} (ax^{(i)} + b - y^{(i)})$$

$$\frac{\partial J(a, b)}{\partial b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$
- Algorithme de Gradient Descent :

$$a = a - \alpha \frac{\partial J(a, b)}{\partial a}$$

$$b = b - \alpha \frac{\partial J(a, b)}{\partial b}$$

Matrix notation with Gradient Descent

Initiation au Machine Learning

6. Régression Linéaire (2/2)



<https://www.youtube.com/watch?v=8Y3r7F47Xfo> (Machine Learnia)

Putting it altogether!

- **Dataset :** (x, y) avec m exemples

- **Modèle :** $f(x) = ax + b$

- **Fonction Coût :** $J(a, b) = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$

- **Gradients :**

$$\frac{\partial J(a, b)}{\partial a} = \frac{1}{m} \sum_{i=1}^m x^{(i)} (ax^{(i)} + b - y^{(i)})$$

$$\frac{\partial J(a, b)}{\partial b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$

- **Algorithme de Gradient Descent :**

$$a = a - \alpha \frac{\partial J(a, b)}{\partial a}$$

$$b = b - \alpha \frac{\partial J(a, b)}{\partial b}$$

Régression Linéaire: Résumé des équations

- **Dataset :** (x, y) avec m exemples, n variables

$$X = \begin{bmatrix} x^{(1)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \quad \theta = \begin{bmatrix} a \\ b \end{bmatrix}$$

$m \times (n+1) \qquad m \times 1 \qquad (n+1) \times 1$

- **Modèle :** $F = X \cdot \theta \qquad m \times 1$

- **Fonction Coût :** $J(\theta) = \frac{1}{2m} \sum (X \cdot \theta - Y)^2 \qquad 1 \times 1$

- **Gradients :** $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T \cdot (X \cdot \theta - Y) \qquad (n+1) \times 1$

- **Gradient Descent :** $\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \quad \text{(gear icon)} \qquad (n+1) \times 1$

<https://machinelearning.com/>

Programming GD in Python

Assignment to do in lab

Initiation au Machine Learning

8. Numpy Regression Linéaire



<https://www.youtube.com/watch?v=vG6tDQc86Rs> (Machine Learnia)

Code on github: <https://github.com/MachineLearnia/Regression-lineaire-numpy/blob/master/R%C3%A9gression%20Lin%C3%A9aire%20Numpy.ipynb>

Programming GD in Python

Assignment to do in lab

Initiation au Machine Learning

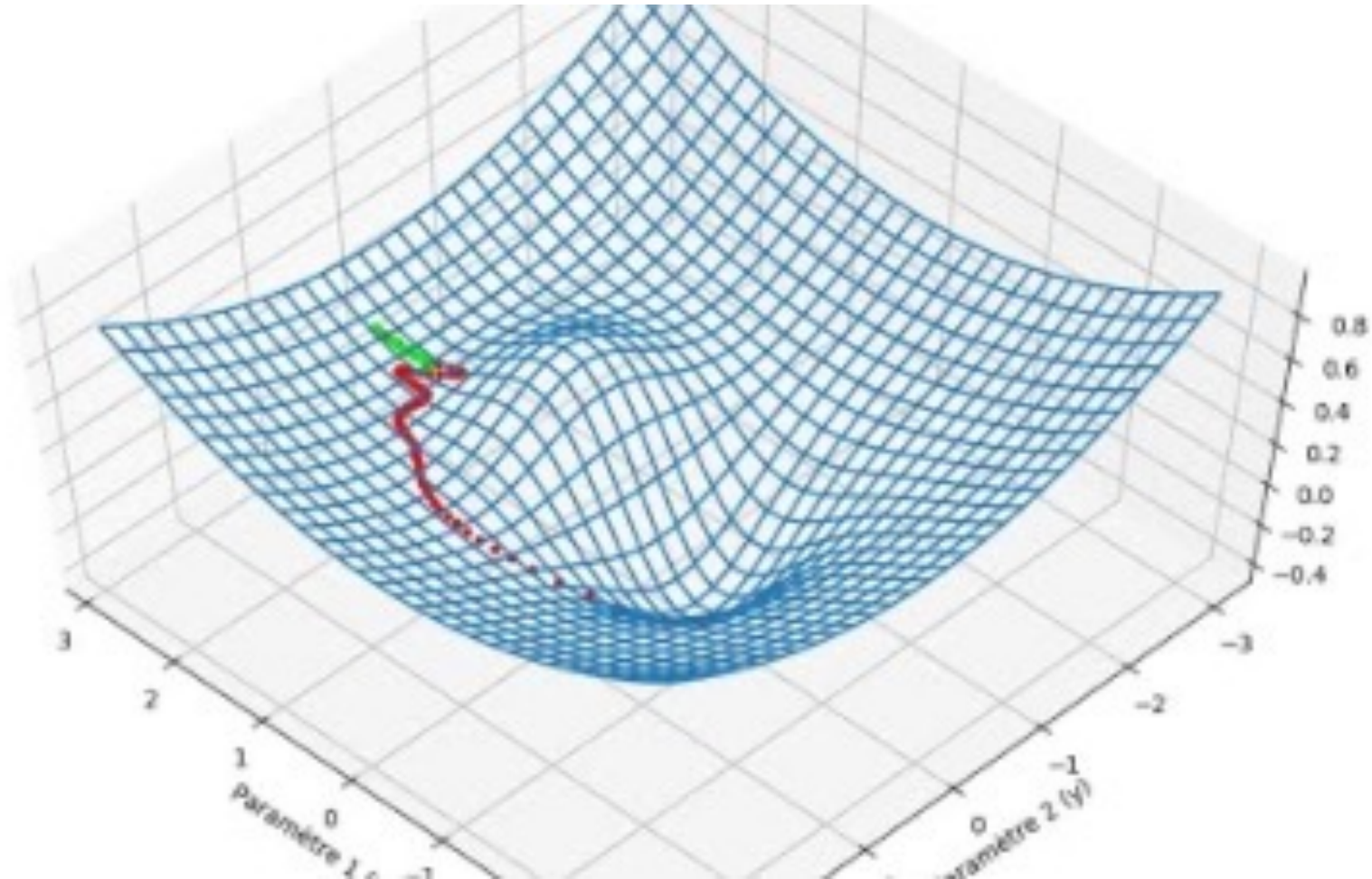
9. Régression Linéaire Multiple



<https://www.youtube.com/watch?v=cpltYCNLI0> (Machine Learnia)

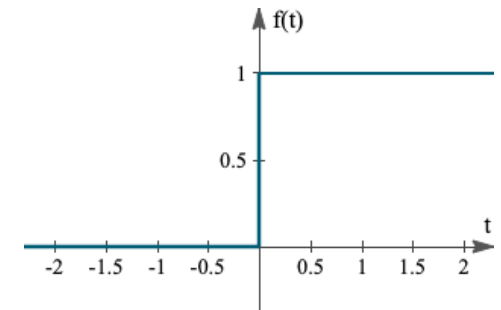
Code on github: <https://github.com/MachineLearnia/Regression-lineaire-numpy/blob/master/R%C3%A9gression%20Lin%C3%A9aire%20Multiple.ipynb>

Going deeper in GD



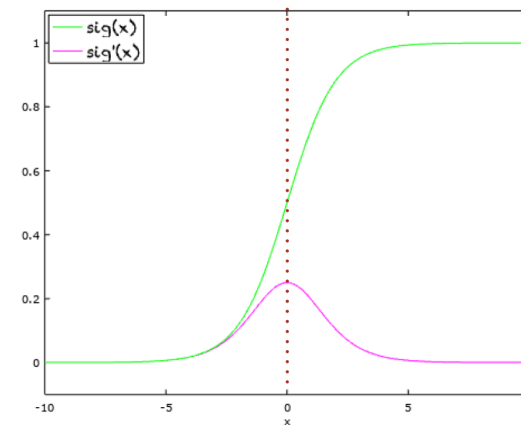
Gradient Descent & output functions

- ⦿ The only non-linear function that can be used as an activation function in a neural network is one which is monotonically increasing. So for example, $\sin(x)$ or $\cos(x)$ cannot be used as activation functions.
- ⦿ Also, the activation function should be defined everywhere and should be continuous everywhere in the space of real numbers. The function is also required to be differentiable over the entire space of real numbers
- ⦿ For instance Heaviside is not for $x=0$
- ⦿ Typically a back propagation algorithm uses gradient descent to learn the weights of a neural network. To derive this algorithm, the derivative of the activation function is required



Example: the Logistic function

- ⦿ The graph of sigmoid function is an S-shaped curve as shown by the green line in the graph below. The figure also shows the graph of the derivative in pink color. The expression for the derivative, along with some important properties are shown on the right
- ⦿ The fact that the sigmoid function is monotonic, continuous and differentiable everywhere, coupled with the property that its derivative can be expressed in terms of itself, makes it easy to derive the update equations for learning the weights in a neural network when using back propagation algorithm



Plot of $\sigma(x)$ and its derivate $\sigma'(x)$

Domain: $(-\infty, +\infty)$
Range: $(0, +1)$
 $\sigma(0) = 0.5$

Other properties

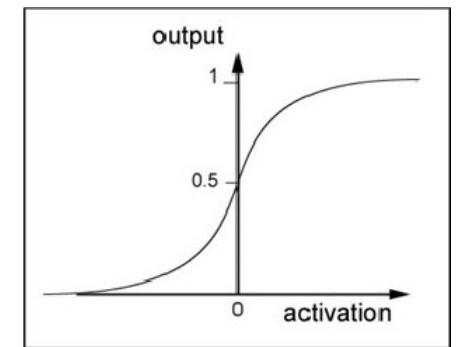
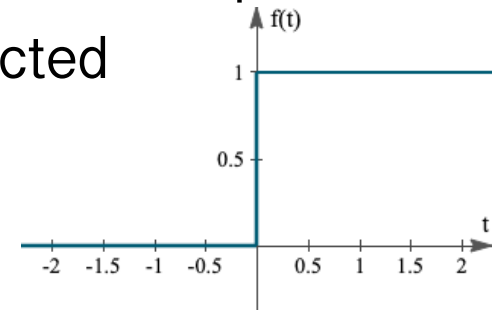
$$\sigma(x) = 1 - \sigma(-x)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

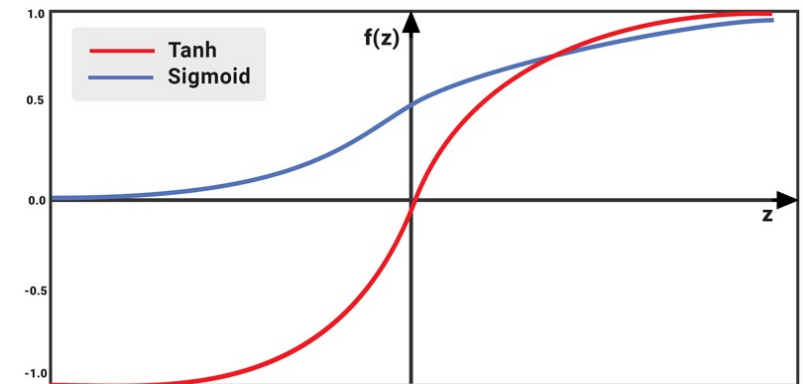
Choosing a right activation function (1)

- Closely related to the efficiency of the learning phase, the choice of the activation function can have an important impact
- For instance, the convergence of GD can be affected
- For instance, the derivative should provide some "information". Heaviside derivative is mainly 0 so GD will not perform well
- The Sigmoid function can "kill" gradient: Sigmoid neurons get saturated on the boundaries and hence the local gradients at these regions is almost zero (vanishing gradient problem)
- We also need to avoid saturation because, if the initial weights are too large the network will hardly learn



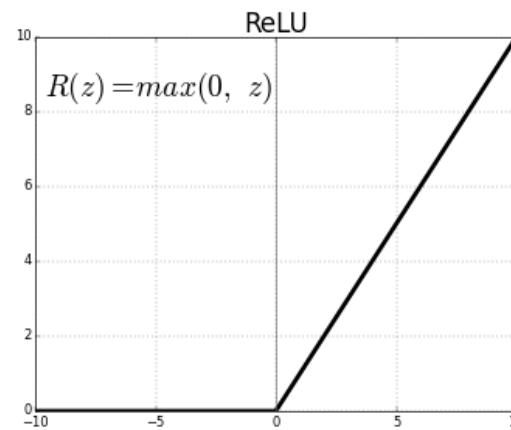
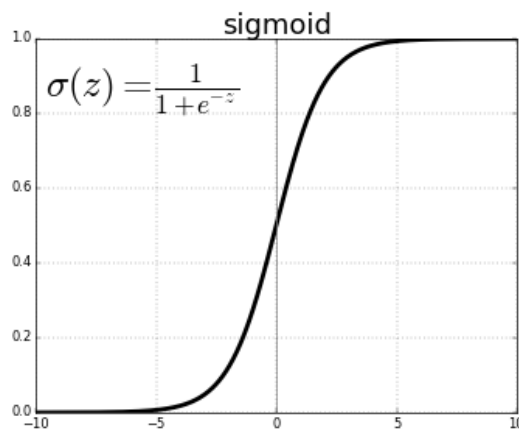
Choosing a right activation function (2)

- ⦿ Sigmoid is also non zero-centered outputs: output after applying sigmoid is always positive so the gradient on the weights during backpropagation will always be either positive or negative which makes optimization harder
- ⦿ The Tanh or hyperbolic tangent Activation Function can somehow be similar to Sigmoid and saturates at large positive and negative
- ⦿ However, its output is always zero-centered which helps since the neurons in the later layers of the network would be receiving inputs that are zero-centered. Hence, in practice, tanh activation functions are preferred in hidden layers over sigmoid



The ReLU function

- ⦿ Rectified Linear Unit (ReLU) become popular as it was found that it greatly accelerates the convergence of optimization process compared to Sigmoid or Tanh activation functions



$$\text{ReLU}(z) = \max(0, z)$$

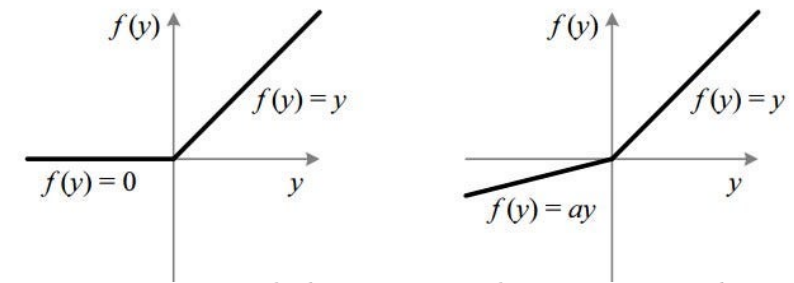
- ⦿ It is non-linear(although it's acts like a linear function for $x > 0$)
- ⦿ ReLU is cheap to compute so model takes less time to run
- ⦿ At a time only a few neurons are activated making the network sparse and usually efficient

ReLU is not perfect neither

- Because of the horizontal line in ReLU (for negative x), the gradient can go towards 0. Those neurons which go into that state will stop responding to variations in error/ input (simply because gradient is 0, nothing changes). This is called **dying ReLU problem**. This problem can cause several neurons to just die and not respond making a substantial part of the network passive

- Leaky ReLU

- We saw that for values less than 0, the gradient is 0 which results in “Dead Neurons” in those regions

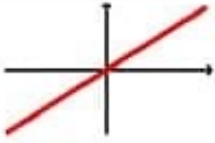
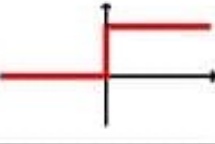
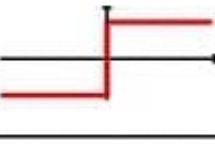

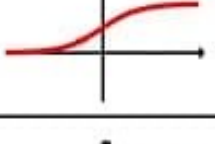




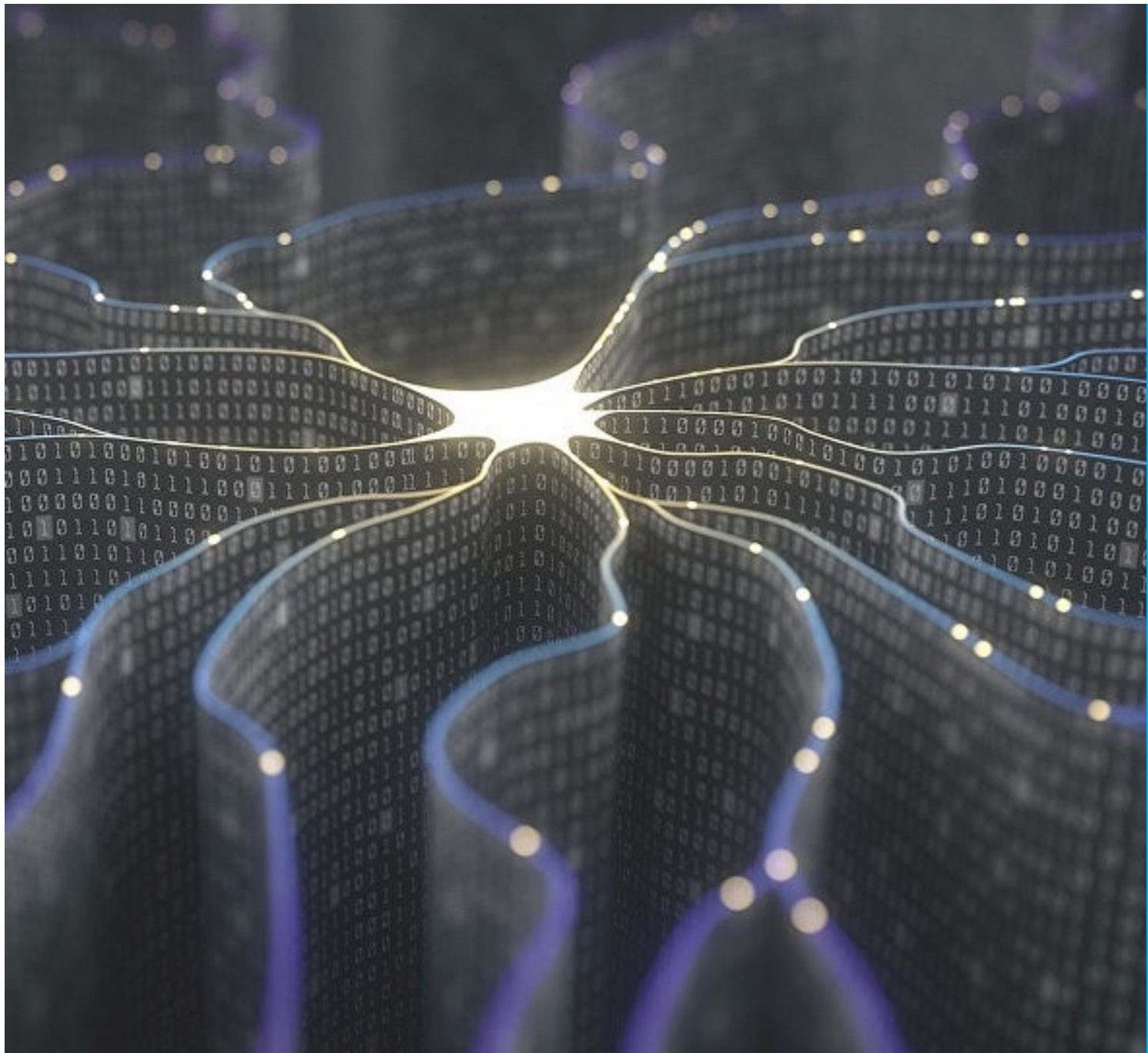
$$\text{LeakyReLU}(z) = \max(0.01 * z, z)$$

- Leaky ReLU addresses this problem: instead of defining values less than 0 as 0, we instead define negative values as a small linear combination of the input. The small value commonly used is 0.01.

Synthesis

<https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron>

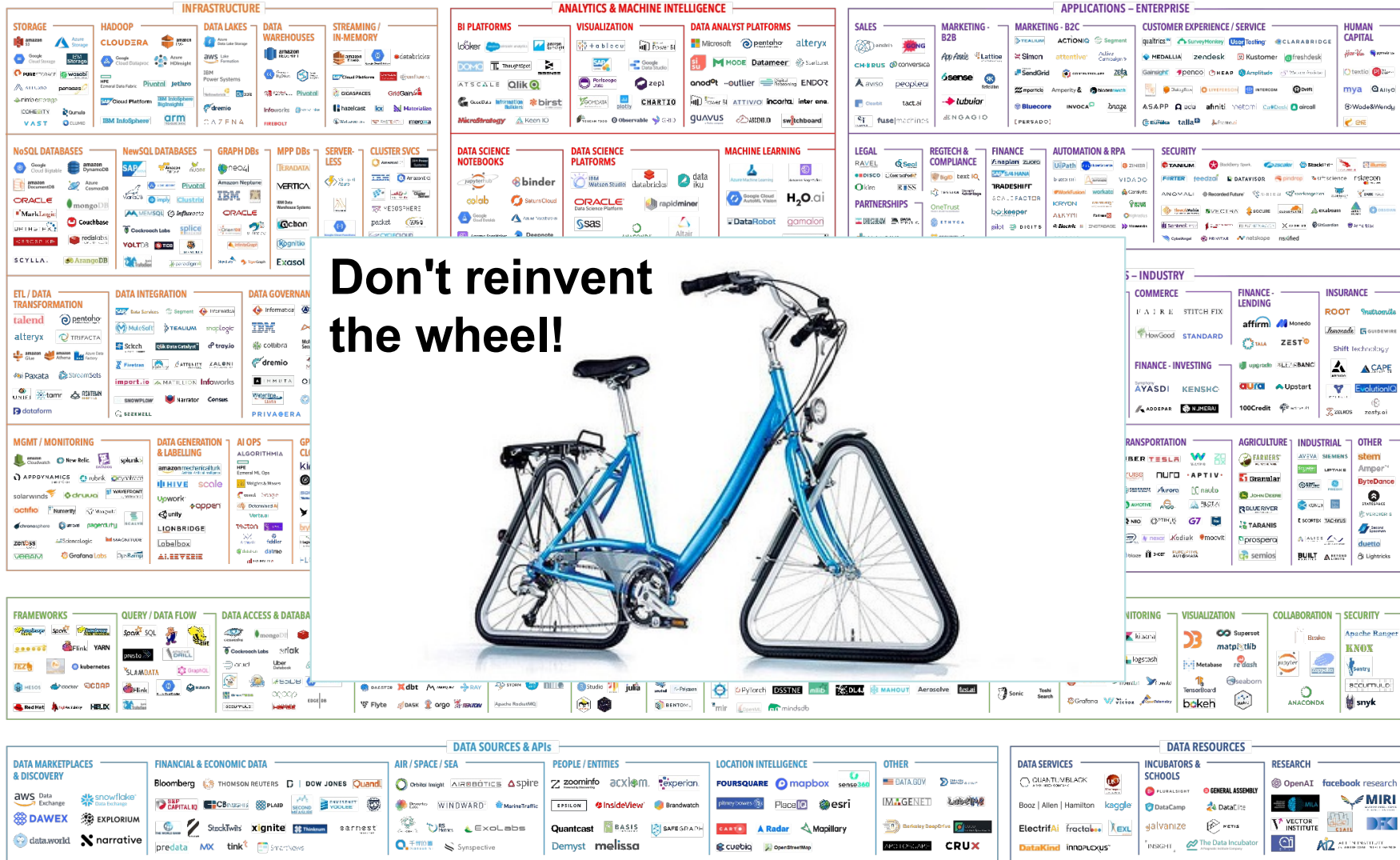
Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -1/2 \\ z + 1/2 & -1/2 \leq z \leq 1/2 \\ 1 & z \geq 1/2 \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	



TOOLS & LIBS FOR AI

The BigData & AI Landscape

DATA & AI LANDSCAPE 2020



Pr. Congduc Pham
http://www.univ-pau.fr/~cpham

Package: pandas

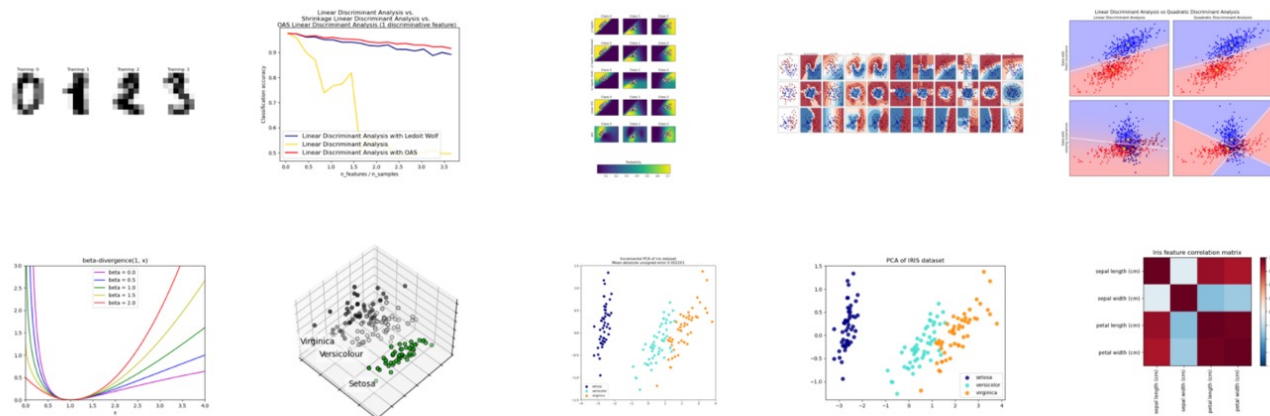
- ⦿ Python Data Analysis Library
- ⦿ pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language
- ⦿ pandas provides fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive
- ⦿ it aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.
- ⦿ https://youtu.be/_T8LGqJtuGc

Package: Numpy

- ① <https://numpy.org/>
- ① NumPy is the fundamental package for scientific computing in Python
- ① It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
- ① The Absolute Beginner's Guide:
https://numpy.org/doc/stable/user/absolute_beginners.html

Package: Scikit-Learn

- Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning
- It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities
- https://scikit-learn.org/stable/auto_examples/index.html



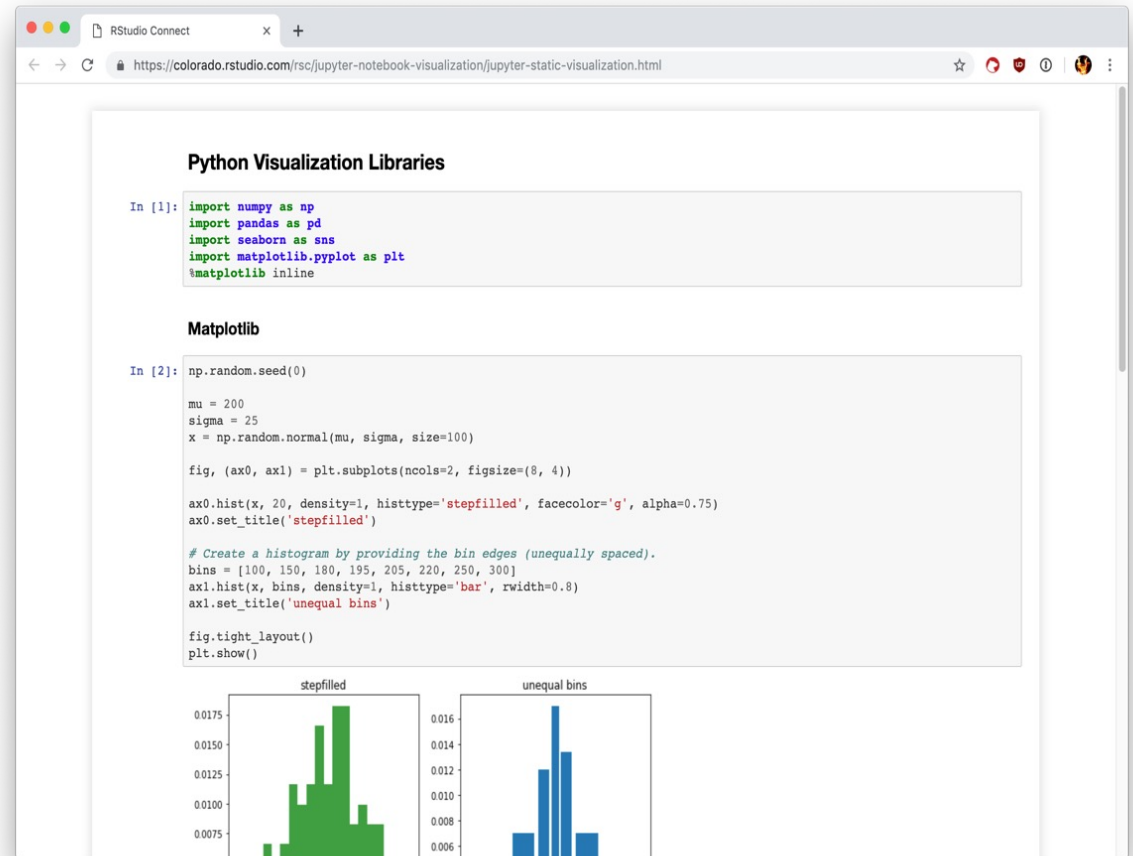
Package: XGBoost

- ⦿ eXtreme Gradient Boosting is an optimized open source implementation of the gradient boosting trees algorithm
- ⦿ Gradient Boosting is a **supervised learning algorithm** whose principle is to combine the results of a set of data and weaker models in order to provide a **better prediction** (regression)
- ⦿ XGBoost includes a large number of hyperparameters which can be modified and tuned for improvement
- ⦿ XGBoost is not part of Scikit-Learn but works perfectly with it
- ⦿ XGBoost behaves remarkably in machine learning competitions!
- ⦿ Source: "XGBoost: The super star of algorithms in ML competition". See examples from <http://aishelf.org/xgboost/>

Jupyter Notebook

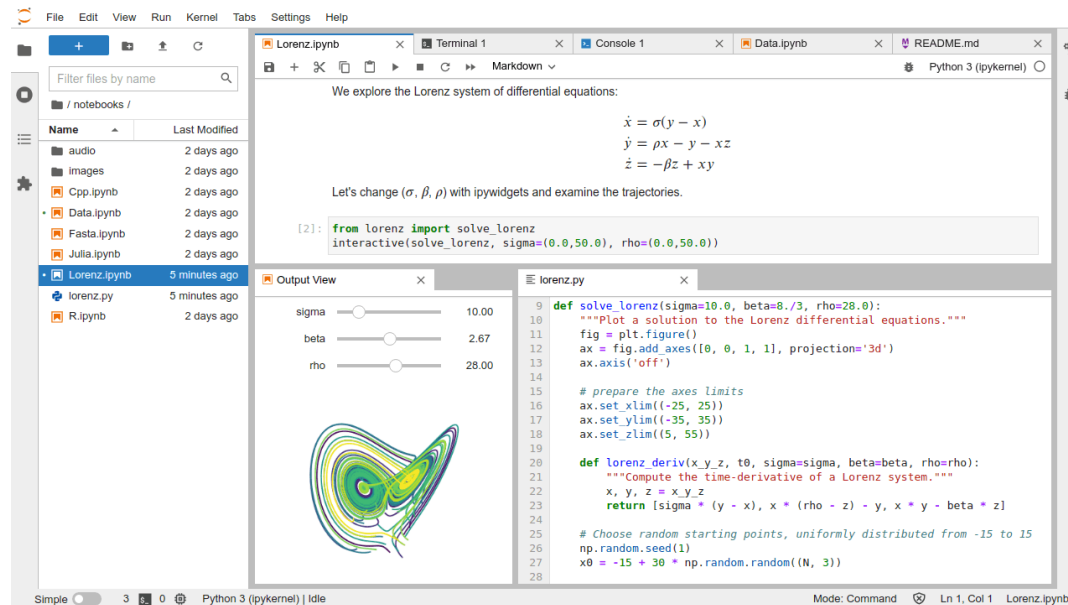
Web application that allows to create and share documents that contain live code, equations, visualizations and narrative text

- ⦿ Data cleaning and transformation
- ⦿ Numerical simulation
- ⦿ Statistical modeling
- ⦿ Data visualization
- ⦿ Machine learning, and more



JupyterLab

- ⦿ Jupyter Notebook only offers a very simple interface using which users can open notebooks, terminals, and text files
- ⦿ Jupyter Lab offers a very interactive interface that includes notebooks, consoles, terminals, CSV editors, markdown editors, interactive maps, and more



The screenshot displays the JupyterLab environment. On the left, a file browser shows a list of notebooks and files. The main workspace is divided into several panes:

- Terminal 1:** Contains the Lorenz system of differential equations:

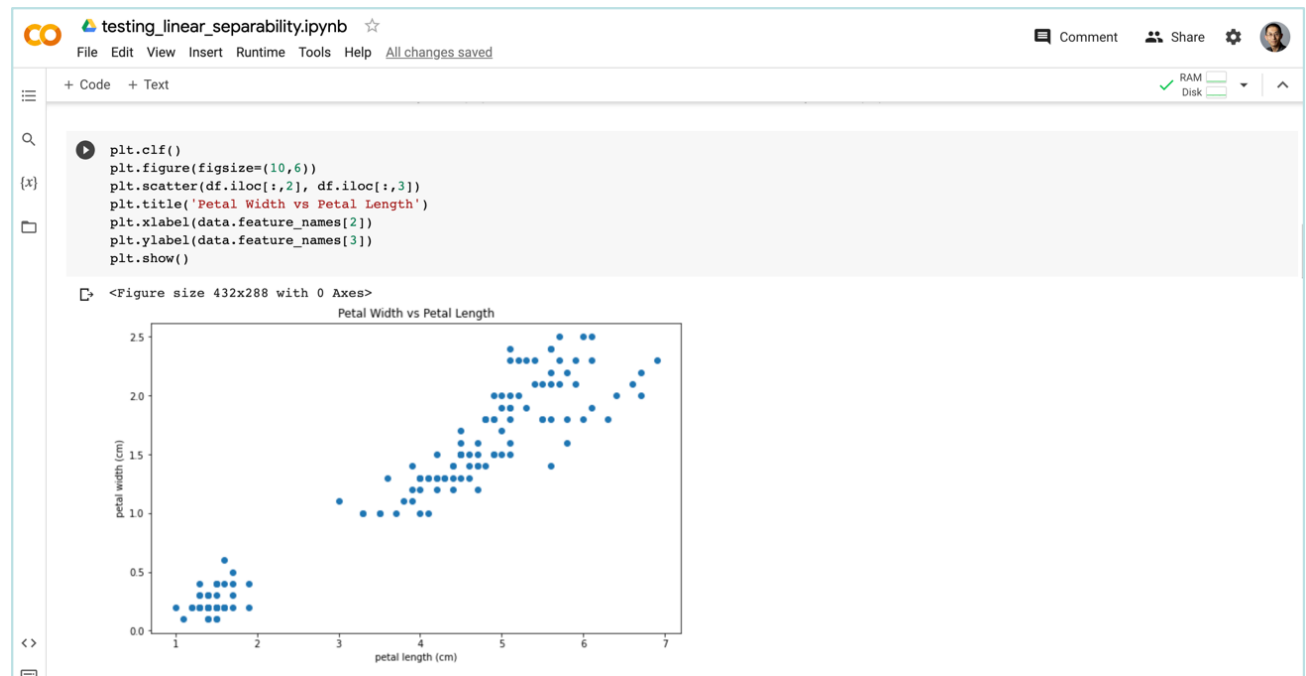
$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned}$$
 Below the equations, it says "Let's change (σ, β, ρ) with ipywidgets and examine the trajectories."
- Code Editor:** Shows a code cell with the following code:


```
[2]: from lorenz import solve_lorenz
      interactive(solve_lorenz, sigma=(0.0,50.0), rho=(0.0,50.0))
```
- Output View:** Features three interactive sliders for parameters:
 - sigma: 10.00
 - beta: 2.67
 - rho: 28.00
 Below the sliders is a 3D plot of the Lorenz attractor, showing its characteristic butterfly shape.
- Code Editor (lorenz.py):** Contains the implementation of the Lorenz system solver and plotter.


```
9 def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x_dot, y_dot, z_dot = x,y,z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```

Colab

- Colab, or "Colaboratory", allows you to write and execute Python in your browser, with
 - Zero configuration required
 - Access to GPUs free of charge
 - Easy sharing



Binder

- ⦿ Colab is an interactive editor environment (allowing you to edit and run your Jupyter notebooks)
- ⦿ Binder is for running Jupyter notebooks as applications

Package: TensorFlow

In mathematics, a tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space. Tensors may map between different objects such as vectors, scalars, and even other tensors.

 Wikipedia
<https://en.wikipedia.org/wiki/Tensor>

[Tensor - Wikipedia](#)

- ⦿ TensorFlow is an end-to-end open source platform for machine learning and deep learning. It is developed by Google.
- ⦿ It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications
- ⦿ "TensorFlow is more of a low-level library whereas Scikit-Learn comes with off-the-shelf algorithms, e.g., algorithms for classification such as SVMs, Random Forests, Logistic Regression, ..."
[<https://stackoverflow.com/questions/61233004>]
- ⦿ <https://www.tensorflow.org/>
- ⦿ <https://www.tensorflow.org/overview>

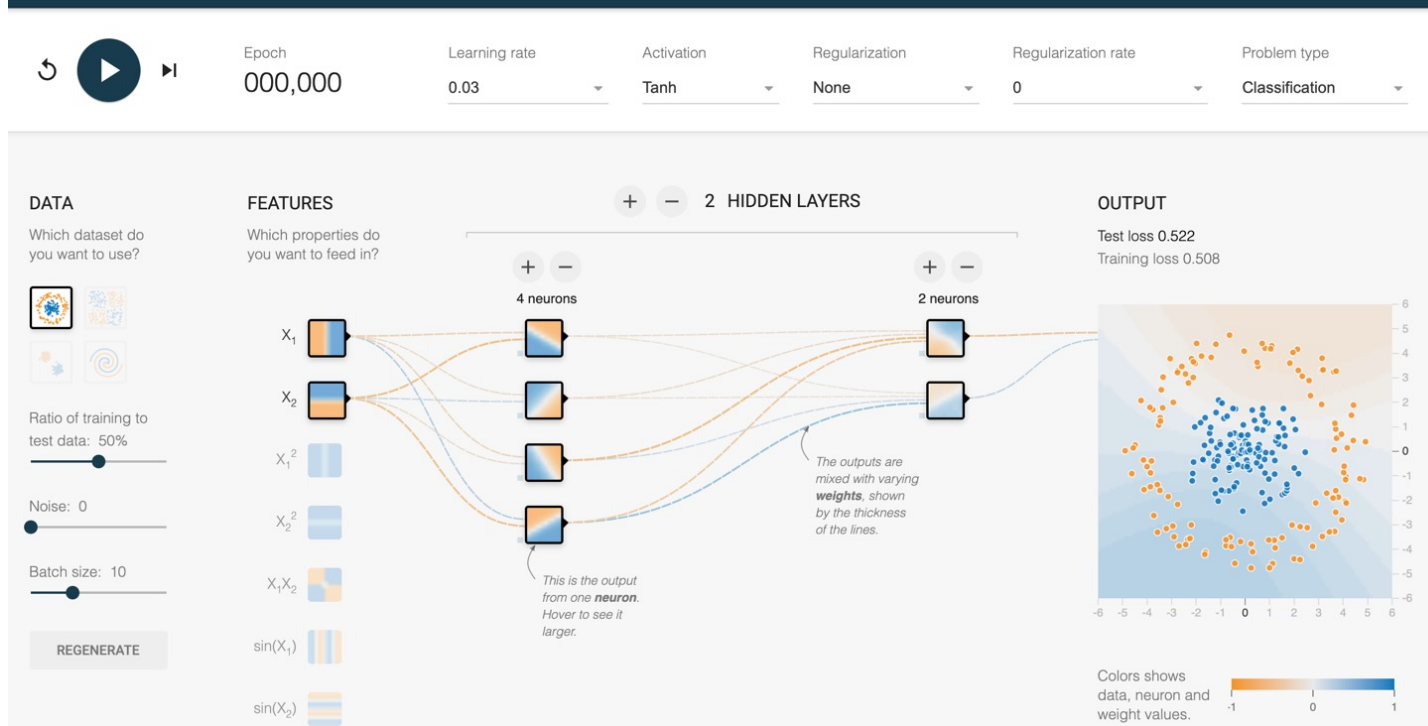


TensorFlow

Want to try DeepLearning?

🕒 <https://playground.tensorflow.org/>

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.



Epoch: 000,000
Learning rate: 0.03
Activation: Tanh
Regularization: None
Regularization rate: 0
Problem type: Classification

DATA: Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE

FEATURES: Which properties do you want to feed in?
 X_1
 X_2
 X_1^2
 X_2^2
 $X_1 X_2$
 $\sin(X_1)$
 $\sin(X_2)$

2 HIDDEN LAYERS
4 neurons
2 neurons

OUTPUT: Test loss 0.522
Training loss 0.508

Colors shows data, neuron and weight values.

Package: Keras

- ① <https://keras.io/> – Deep learning for humans
- ① Keras is an effective high-level neural network API written in Python
- ① This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can run on top of TensorFlow for instance
- ① It also has extensive documentation and developer guides
- ① Keras was adopted and integrated into TensorFlow in mid-2017. Users can access it via the `tf.keras` module. However, the Keras library can still operate separately and independently.

Package: PyTorch

- ① <https://pytorch.org/>
- ② PyTorch is a machine learning framework based on the Torch library used for applications such as computer vision and natural language processing originally developed by Meta AI and now part of the Linux Foundation umbrella [Wikipedia]
- ③ PyTorch is used for many deep learning projects today, and its popularity is increasing among AI researchers